VŠB-Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra telekomunikační techniky

Vytvoření grafického uživatelského rozhraní v prostředí MATLAB pro komunikaci a zpracování dat z osciloskopu Tektronix TDS 2002B

Graphical user interface in MATLAB for communication with the Tektronix TDS 2002B oscilloscope

Akademický rok 2009/2010 Bc. Roman Koutný

Zadání

Téma: Vytvoření grafického uživatelského rozhraní v prostředí MATLAB pro komunikaci a zpracování dat z osciloskopu Tektronix TDS 2002B

Diplomant: Bc. Roman Koutný

Vedoucí diplomové práce: Ing. Jan Skapa, Ph.D.

Akad. rok: 2009/2010

Obor: Telekomunikační technika

Zásady pro vypracování:

1. Popis komunikace osciloskopu Tektronix TDS 2002B s prostředím MATLAB
2. Vytvoření grafického uživatelského rozhraní (GUI) pro ovládání osciloskopu a zpracování dat
3. Vytvoření návodu pro práci s GUI

Seznam doporučené odborné literatury:

- ZAPLATÍLEK, K. - DOŇAR, B.: *Matlab – tvorba uživatelských aplikací*, 1.vyd. BEN, Praha, 2004.215 s. ISBN 80-7300-133-0
- Webové stránky výrobce osciloskopu. Dostupné z WWW: <http://www2.tek.com/cmswpt/psdetails.lotr?ct=PS&cs=psu&ci=13295&lc=EN>
- MATLAB Central - An open exchange for the MATLAB and Simulink user community. Dostupné z WWW: < http://www.mathworks.com/matlabcentral/>

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Orlové-Lutyni dne 14. 4. 2010 Bc. Koutný Roman

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu práce, Ing. Janu Skapovi, Ph.D., za trpělivost, cenné rady, připomínky a náměty, kterými mě při realizaci mé práce motivoval. Pomáhal mi tak mou práci rozvíjet a zdokonalovat. Rovněž mu děkuji za pomoc při zajišťování technických podmínek pro realizaci práce a konzultace v období mimo výuku.

Abstrakt a klíčová slova

Abstrakt:

V této práci je popsána základní problematika spojená s ovládáním osciloskopu Tektronix TDS 2002B a následně zpracováním naměřených údajů při využití matematického softwaru MATLAB. Spojení počítače s osciloskopem je realizováno přes rozhraní USB. Na monitoru obslužného počítače je pomocí grafického uživatelského prostředí GUIDE vytvořena obslužná aplikace. Uživateli je tak umožněno pomocí myši, případně klávesnice ovládat, nastavovat a měnit základní vlastnosti měřicího přístroje a ovlivnit tak způsob zobrazení měřené veličiny.

Klíčová slova:

GUI, GUIDE, MATLAB, osciloskop, Tektronix

Abstract:

In this work is described basic dilemma connected with operating of oscilloscope Tektronix TDS 2002B and consequently processing measured data using mathematic software MATLAB. Connection PC with oscilloscope is realized over USB interface. On service PC monitor is created service application using of graphic user's environment GUIDE. User is allowed using of mice, eventually keyboard take control of, adjust, change basic characteristics of measuring instrument, and influence so mode of representation measured quantity.

Keywords:

GUI, GUIDE, MATLAB, oscilloscope, Tektronix

Obsah

Seznam použitých zkratek a symbolů

Zkratka	Význam
FFT	**F**ast **F**ourier **T**ransform (rychlá Fourierova transformace) je efektivní algoritmus pro výpočet diskrétní Fourierovy transformace. FFT je velmi důležitá v mnoha oblastech, od digitálního zpracování signálu a řešení parciálních diferenciálních rovnic až po rychlé násobení velkých celých čísel.
GUI	**G**raphical **U**ser **I**nterface (grafické uživatelské rozhraní) je uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků. Na monitoru počítače jsou zobrazena okna, ve kterých programy zobrazují svůj výstup. Uživatel používá klávesnici, myš a grafické vstupní prvky jako jsou menu, ikony, tlačítka, posuvníky, formuláře a podobně.
GUIDE	grafické uživatelské prostředí programu MATLAB, které poskytuje sadu nástrojů pro vytvoření grafických uživatelských rozhraní (GUI). Tyto nástroje usnadňují postup tvorby i naprogramování GUI.
MATLAB	programové prostředí a skriptovací programovací jazyk pro vědeckotechnické numerické výpočty, modelování, návrhy algoritmů, počítačové simulace, analýzu a prezentaci dat, měření a zpracování signálů, návrhy řídicích a komunikačních systémů. Název MATLAB vznikl zkrácením slov **MAT**rix **LAB**oratory, což odpovídá skutečnosti, že klíčovou datovou strukturou při výpočtech v MATLABu jsou matice. Vlastní programovací jazyk vychází z jazyka Fortran.
USB	**U**niversal **S**erial **B**us **(**univerzální sériová sběrnice) - způsob připojení periférií k počítači. Nahrazuje dříve používané způsoby připojení (sériový, paralelní port, PS/2 apod.) pro běžné druhy periférií - tiskárny, myši, klávesnice, fotoaparáty, modemy, externí disky apod.

VISA **V**irtual **I**nstrument **S**tandard **A**rchitecture je standard definovaný firmami Agilent Technologies a National Instruments pro komunikaci s nástroji, bez ohledu na rozhraní. Pomocí VISA standard je podporováno rozhraní GPIB, VXI, GPIB-VXI, TCP/IP, USB, RSIB a sériové rozhraní.

Symbol	**Význam**
MHz	Hertz [Hz] je jednotka frekvence signálu, MHz = 10^6 Hz
GS/s	Sample/sekunda [S/s] definuje počet vzorků nasnímaných za jednotku času GS = 10^9 S [Sample]
DC	**D**irect **C**urrent (stejnosměrná vazba), propouští všechny složky signálu.
AC	**A**lternating **C**urrent (střídavá vazba), blokuje stejnosměrnou složku a potlačuje signály s kmitočtem pod 10 Hz.
GND	**G**rou**nd** - zem
BW Limit	**B**and**w**idth Limit (kmitočtový rozsah), obvykle určený poklesem o 3 dB

1. Úvod do problematiky

1.1 Obecný popis osciloskopu Tektronix

Tektronix TDS 2002B [1] je dvoukanálový osciloskop s barevným LCD displejem, pamětí a schopností měřit v kmitočtovém pásmu do 60 MHz, maximální vzorkovací rychlostí 1 GS/s. Automaticky lze osciloskopem měřit 11 různých parametrů. Současně lze v pravé části displeje přístroje zobrazit hodnoty až pěti z následujících veličin, všechny měřitelné veličiny jsou přehledně uvedeny v tabulce č. 1 [6].

Tabulka č. 1: Druhy parametrů měřitelných osciloskopem Tektronix TDS 2002B

Měřená veličina	**Popis**
Freg	Kmitočet zobrazeného signálu (z naměřené doby první periody).
Period	Doba první periody.
Mean	Střední hodnota napětí z celého zaznamenaného průběhu.
Pk-Pk	Absolutní hodnota rozdílu mezi maximální a minimální hodnotou celého průběhu.
Cyc RMS	Efektivní hodnota první úplné periody průběhu.
Min	Minimální hodnota ze všech 2 500 zaznamenaných hodnot.
Max	Maximální hodnota ze všech 2 500 zaznamenaných hodnot.
Rise Time	Doba mezi přechodem 10 % a 90 % úrovně první vzestupné hrany průběhu.
Fall Time	Doba mezi přechodem 10 % a 90 % úrovně první sestupné hrany průběhu.
Pos Width	Doba odpovídající přechodu 50 % úrovně na první vzestupné a následující sestupné hraně průběhu (šířka kladného impulzu).
Neg Width	Doba odpovídající přechodu 50 % úrovně na první sestupné a následující vzestupné hraně průběhu (šířka záporného impulzu).
None	Žádné měření se neprovádí.

V tabulce č. 2 [6] jsou uvedeny základní možnosti nastavení osciloskopu při měření veličin osciloskopem.

Tabulka č. 2: Základní možnosti nastavení měření

Parametr	**Nastavení**	**Vysvětlení**
Coupling (Vazba)	DC	Umožňuje průchod stejnosměrné i střídavé složky vstupního signálu.
	AC	Nepropouští stejnosměrnou složku vstupního signálu a potlačuje signály s kmitočtem pod 10 Hz.
	GND	Odpojuje vstupní signál.
BW Limit	20 MHz[1]	Omezuje šířku pásma a tím snižuje šum v zobrazeném průběhu. Filtruje signály, aby se omezil šum a další nechtěné vysokofrekvenční komponenty signálu.
	Off	
Volts/Div	Coarse	Volí rozlišení ovladače VOLTS/DIV (Volt /dílek na zobrazovacím rastru). Hodnotou Coarse (Hrubě) se definují stupně 1 – 2 – 5.
	Fine	Položkou Fine (Jemně) se mění rozlišení po malých krocích mezi polohami definovanými hodnotami Coarse.
Probe (Sonda)	1x 10x 100x 1000x	Nastavení přepínače sondy (zobrazení ve vertikálním směru).
Invert	On	Zapnutí a vypnutí invertovaného zobrazení průběhu.
	Off	

1.2 MATLAB

MATLAB [5] je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, měření a zpracování signálů, návrhy řídicích a komunikačních systémů. MATLAB je nástroj jak pro pohodlnou interaktivní práci, tak i pro vývoj širokého spektra aplikací.

[1] U sondy 1x je šířka pásma omezena na 7 MHz.

Výpočetní systém MATLAB se během uplynulých let stal celosvětovým standardem v oblasti technických výpočtů a simulací nejen ve sféře vědy, výzkumu a průmyslu, ale i v oblasti vzdělávání. MATLAB je považován za přelom nejen z hlediska rozsahu, integrace a kvality produktu, ale především z hlediska vztahu k uživateli a jeho pohodlí při práci s programem.

MATLAB poskytuje svým uživatelům především mocné grafické a výpočetní nástroje, ale rovněž rozsáhlé knihovny funkcí spolu s výkonným programovacím jazykem čtvrté generace. Knihovny jsou svým rozsahem využitelné prakticky ve všech oblastech lidské činnosti. Díky své architektuře je MATLAB určen zejména těm, kteří potřebují řešit početně náročné úlohy a přitom nechtějí nebo nemají čas zkoumat matematickou podstatu problémů.

Grafika v MATLABu umožňuje snadné zobrazení a prezentaci výsledků získaných výpočtem. Je možné vykreslit různé druhy grafů - dvourozměrné pro funkce jedné proměnné, třírozměrné pro funkce dvou proměnných, histogramy a další. Všem grafickým objektům je možné téměř libovolně měnit vzhled, a to jak při jejich vytváření, tak po jejich nakreslení.

Otevřená architektura MATLABu vedla ke vzniku knihoven funkcí, nazývaných toolboxy, které rozšiřují použití programu v příslušných vědních a technických oborech. Tyto knihovny, navržené a v jazyce MATLABu napsané nejvýznačnějšími světovými odborníky, nabízejí předzpracované specializované funkce, které je možno rozšiřovat, modifikovat, anebo jen čerpat informace z přehledně dokumentovaných algoritmů.

Pro potřebu zadání této práce využijeme vlastnosti MATLABu, které nám (uživatelům osciloskopu Tektronix) umožní získávat a analyzovat měřená data, graficky je vizualizovat, provádět vlastní měření, vytváření sestav pro jednoznačný popis měření příslušné veličiny.

Systémové požadavky nutné pro úspěšné připojení počítače s osciloskopem [2] (Tektronix):

1. osciloskop s vhodným rozhraním (např. USB 2.0), podporující nástroj vzdálené komunikace,

2. programy MATLAB a MATLAB Instrument Control Toolbox nainstalované v počítači,

3. software VISA nainstalovaný v počítači,

4. ovladač MATLAB vhodný pro požadovaný přístroj (osciloskop). Veškeré dostupné ovladače pro osciloskopy Tektronix a program MATLAB jsou online ke stažení webové stránce firmy MathWorks [4]: http://www.mathworks.com/tektronix.

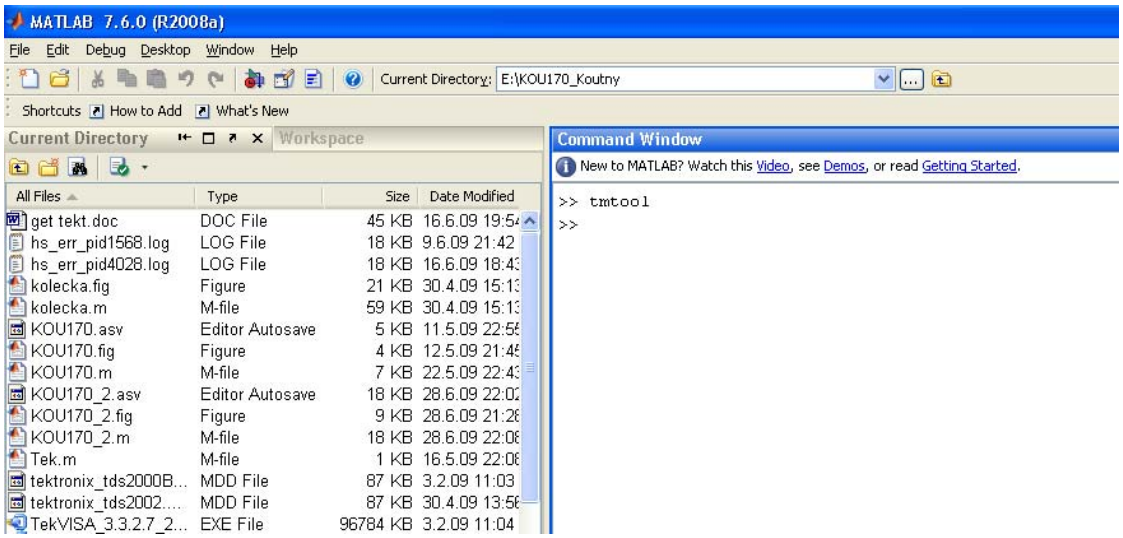
1.3 Propojení počítače a osciloskopu Tektronix

Propojení s jinými zařízeními umožňuje osciloskopu port USB 2.0, který se nachází na zadní straně přístroje (komunikuje s počítačem nebo umožňuje přímý tisk na tiskárnu s rozhraním USB) nebo USB Host Port na předním panelu (umožňuje napojení flash disk). Pro vytvoření jednoduchých i složitých měřicích utilit, usnadnění ovládání, vzdálenou správu osciloskopu, správu komunikace mezi počítačem a osciloskopem je možno použít software více společností. Náplní této práce je vytvoření komunikační utility při využití programového vybavení firmy MathWorks, konkrétně MATLABu (použitá verze 2008a), a to právě s ohledem jeho masivního rozšíření.

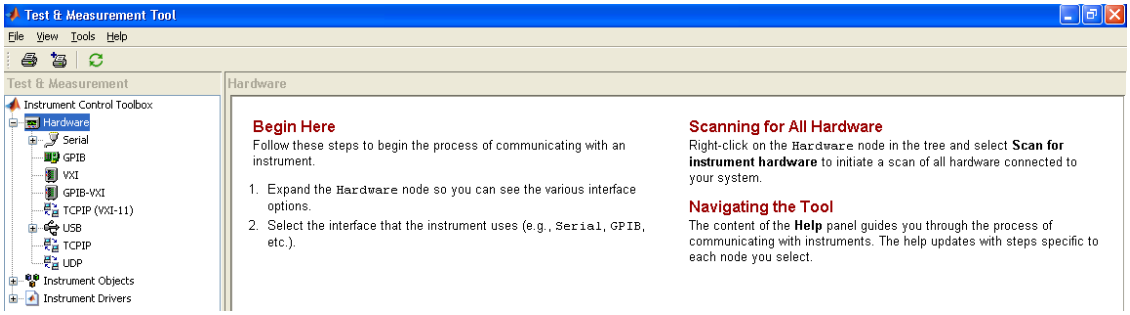
Příprava systému pro připojení osciloskopu k počítači [2]:

Krok 1: Po spuštění MATLABu zadáme příkaz k zahájení utility Test and Measurement Tool:

`tmtool,` jak je patrno z obrázků č. 1 a č. 2.



Obr. 1: Základní okno programu MATLAB



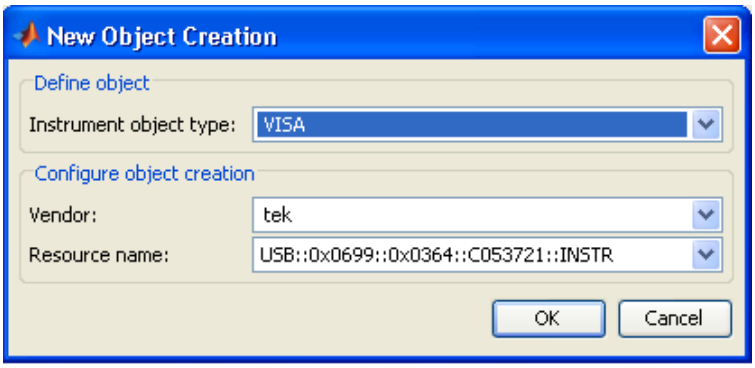
Obr. 2: Okno utility Test and Measurement Tool

Test and Measurement Tool je grafické uživatelské rozhraní pro nastavení a kontrolu nástrojů programu MATLAB.

Krok 2: Připojíme osciloskop k počítači přes rozhraní USB. Ve výchozím nastavení je většina osciloskopů Tektronix nastavena na automatickou detekci připojení k počítači. Následně je zapotřebí zkontrolovat uživatelské rozhraní obslužného software standardu VISA a ověřit, zda počítač s osciloskopem komunikuje. Při komunikaci s počítačem je nezbytné vždy uvádět úplné USB jméno (např. USB0:: 1689:: 871:: C010151:: 0:: INSTR). V případě, že osciloskop není počítačem detekován automaticky, je zapotřebí opětovně zvolit možnost `Refresh All` nebo `Rescan`.

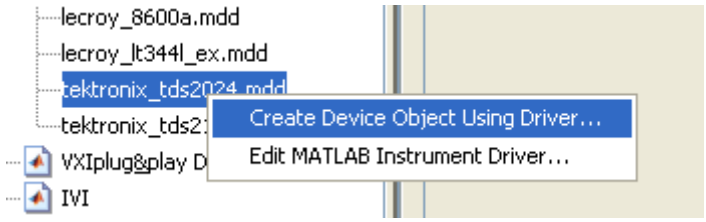
Krok 3: MATLAB s Instrument Control Toolbox podporuje intuitivní způsob řízení, který nám umožní komunikovat s osciloskopem bez znalosti základních ovladačů. Chceme-li zjistit, zda ovladače předmětného osciloskopu jsou již v našem počítači instalovány, klikneme pravým tlačítkem myši na `Instrument Drivers` a vyberme `Scan for Instrument Drivers`. Pokud nebude obslužným softwarem automaticky nainstalován odpovídající ovladač, zkontrolujeme dostupnost požadovaného ovladače na internetových stránkách firmy MathWorks [4]: http://www.mathworks.com/tektronix. Pokud se nám na internetových stránkách nepodaří najít ovladač s požadovaným názvem (shodný s názvem přístroje), je firmou Mathworks doporučeno vyzkoušet ovladač pro funkčně podobné zařízení. V naprosté většině případů bude zařízení pracovat správně.

Krok 4: Přejdeme k určení rozhraní Interface Object. Klikneme pravým tlačítkem myši a zvolíme možnost `Create New Interface Object`. Vybereme typ měřicího přístroje, standard VISA a také název zdroje – opět musíme zadat úplné „USB jméno“ (viz Krok 2). Proces konfigurace rozhraní je patrný z obrázku č. 3.



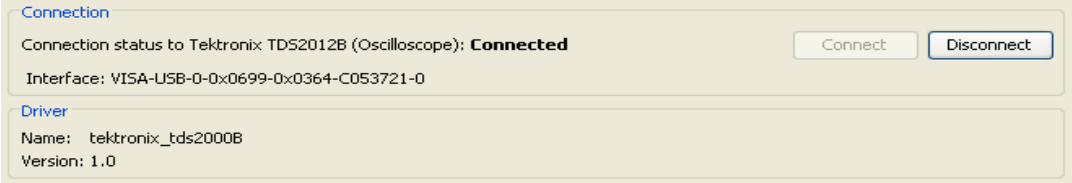
Obr. 3: Dialog určení rozhraní pro zařízení

Krok 5: Pravým tlačítkem myši klikneme na ovladač určený pro osciloskop a zvolíme možnost `Create Device Object Using Driver`. V následném dialogovém okně vybereme rozhraní objektu, které jsme vytvořili v předcházejícím kroku (krok č. 4) – viz obrázek č. 4.



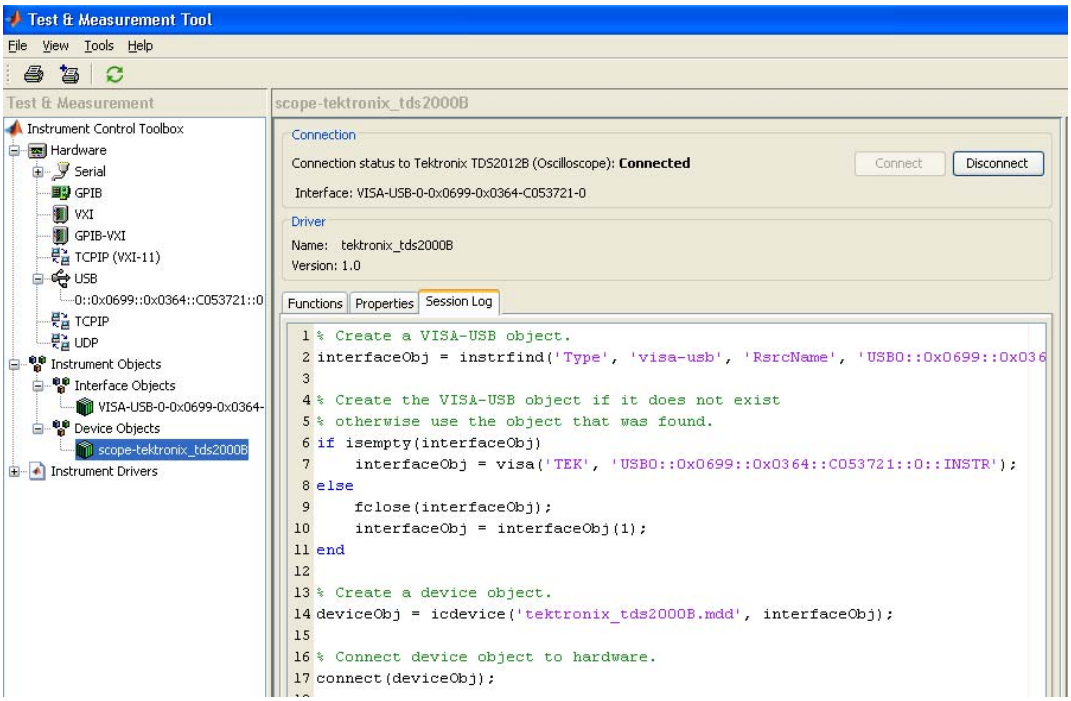
Obr. 4: Výběr ovladače

Krok 6: Vybereme objekt vytvořený v předchozím kroku. Po stisku tlačítka připojení `Connect` v pravém horním rohu, jak je patrno z obrázku č. 5, je počítač připraven zahájit komunikaci se zařízením (osciloskopem Tektronix).



Obr. 5: Zahájení komunikace

V záložce Properties (vlastnosti) můžeme kontrolovat a rovněž měnit základní nastavení a konfiguraci osciloskopu, úplný výpis komunikace s přístrojem (využitelný dále např. pro komunikaci přístroje s PC přes program MATLAB) si můžeme zobrazit v záložce Session Log (záznam práce). Ukázka výpisu záložky Session Log je uveden na obrázku č. 6.



Obr. 6: Výpis záznamu připojení po zahájení komunikace

2. Grafické uživatelské prostředí a rozhraní

GUIDE je grafické uživatelské vývojové prostředí (grafický editor), které poskytuje sadu nástrojů pro vytvoření grafických uživatelských rozhraní (GUI). Tyto nástroje usnadní postup při tvorbě a také při programování GUI. Použitím GUIDE Layout Editoru můžeme vytvořit GUI jednoduše pomocí myši přetažením GUI součástí (například os, panelů, tlačítek, textových polí, posuvníků, zatrhávacích políček) na plochu. V Layout Editoru můžeme dále upravit například rozměr okna GUI, jeho celkový vzhled nebo jen vzhled jednotlivých komponent apod. GUIDE automaticky generuje M–file (soubor s příponou *.m) a soubor figure (soubor s příponou *.fig). Po napsání názvu souboru do příkazového řádku (například `tektronix.m`), lze GUI kdykoli jednoduše spustit.

2.1 Obecný postup při vytváření GUI

Při tvorbě velmi jednoduchých aplikací je možno využít způsob postupného rozšiřování nebo modifikace jednotlivých prvků (případně celých komponent) tak, jak uznáme za vhodné nebo jak se nám projeví jako účelné pro modifikování funkčnosti aplikace [3].

Pro řešení složitějších úkolů respektive při psaní delších programů je výhodné postupovat podle doporučeného postupu. Podle průvodce GUIDE lze postup rozdělit do čtyř kroků:

1. před vlastní tvorbou uživatelského rozhraní je vhodné si nejprve na papír nakreslit, jak si výsledné GUI představujeme:

- vlastní rozvržení jednotlivých prvků na ploše,

- funkce jednotlivých částí (které veličiny budou ovládané, jaké prvky pro jejich ovládání použijeme),

- jakou formu výstupu aplikace požadujeme – text, graf, tisk, uložení, …

2. Na plochu Layout area umístíme příslušné prvky v jednom exempláři, upravíme jejich velikost, rozmístění a zarovnání. Pokud se některý z prvků v GUI vícekrát opakuje, je vhodné použít vždy stejnou velikost. Naklonování objektů provedeme například stiskem levého tlačítka myši na objekt a volbou `Duplicate` nebo klasicky přes menu GUIDE Layout editoru.

3. Vygenerujeme soubory s příponami *.fig (figure) a *.m (soubor MATLAB). Provedeme spuštění *.fig souboru a zkontrolujeme, zda vzhled GUI odpovídá našim představám (nebo zadání). Pokud vzhled není vyhovující, tak znovu editujeme soubor *.fig podle požadavku. V Property Inspectoru můžeme rovněž nastavit počáteční hodnoty a stav proměnných, názvy jednotlivých komponent, jejich barvu, barvu pozadí, jednotky apod.

4. Do vygenerovaného souboru s příponou *.m doplníme zpětné vazby jednotlivých komponent (Callbacky) a ověříme funkci vytvořeného GUI.

2.2 Tvorba GUI pro ovládání Tektronixu

Podle doporučení (obecných zásad) jsem při řešení zadání jako první krok provedl tuto rozvahu:

- jaký vzhled GUI zvolím pro snadné ovládání osciloskopu,

- které z vlastností osciloskopu bude možno programem (pomocí GUI) nastavovat,

- kterými komponenty (rozbalovací menu, posuvník, tlačítko, editační políčko) zajistím ovládání přístroje,

- jakým způsobem zajistím zpětnou kontrolu funkčnosti ovládání osciloskopu (zobrazení měřeného průběhu signálu, uložení naměřených dat).

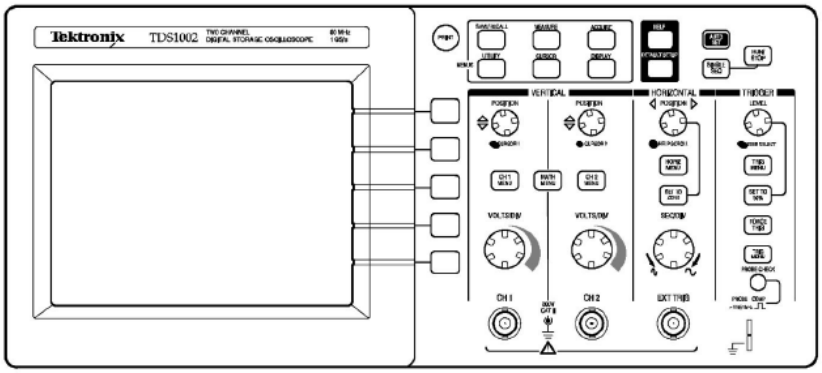
Při volbě vzhledu grafického rozhraní jsem se snažil inspirovat skutečným vzhledem osciloskopu (jeho čelním panelem). I když samozřejmě není možno dodržet shodu vzhledu navrhovaného GUI s osciloskopem Tektronix na 100 %, bylo mým záměrem přiblížit se co možná nejvíce skutečnému vzhledu osciloskopu. Také způsob ovládání vybraných funkcí GUI jsem se snažil přiblížit ovládání osciloskopu tak, aby „komfort“ pro uživatele zůstal zachován.

Základním úkolem při psaní programu pro mne bylo navázání komunikace počítače s osciloskopem, získání dat, kterými lze jednoznačně popsat měřený průběh veličiny a jejich uložení v co možná nejrealističtější podobě. Dalším z úkolů aplikace bylo zajištění možnosti ovládání základních parametrů při zobrazování průběhu veličiny (časová základna signálu, nastavení formátu a rastru zobrazení).

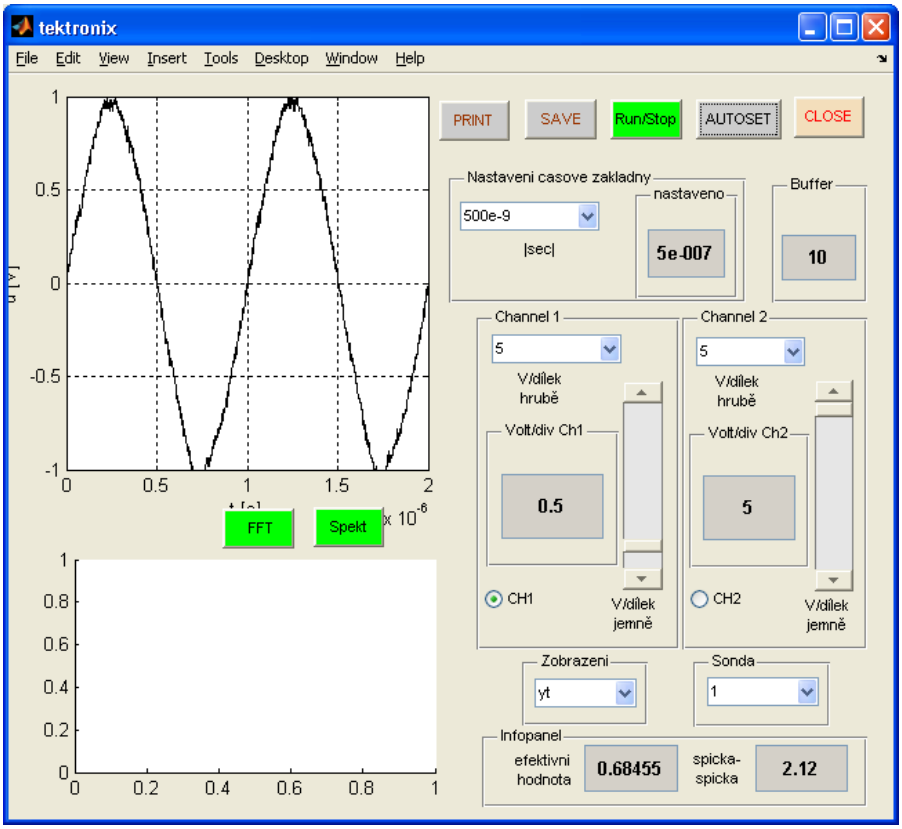
Protože osciloskop Tektronix TDS2002B umožňuje vykreslení Fourierovy transformace signálu, stanovil jsem si jako další z postupných cílů tvorby programu možnost zobrazení průběhu FFT a také vykreslení spektrogramu měřené veličiny.

2.2.1 Rozvržení ovládacích prvků

Jak již bylo dříve zmíněno, vizualizaci GUI jsem pořídil sladění jednotlivých ovládacích prvků se skutečným přístrojem, který je zobrazen na obrázku č. 7. Na obrázku č. 8 je vyobrazen panel vytvořeného GUI.



Obr. 7: Čelní pohled na osciloskop Tektronix

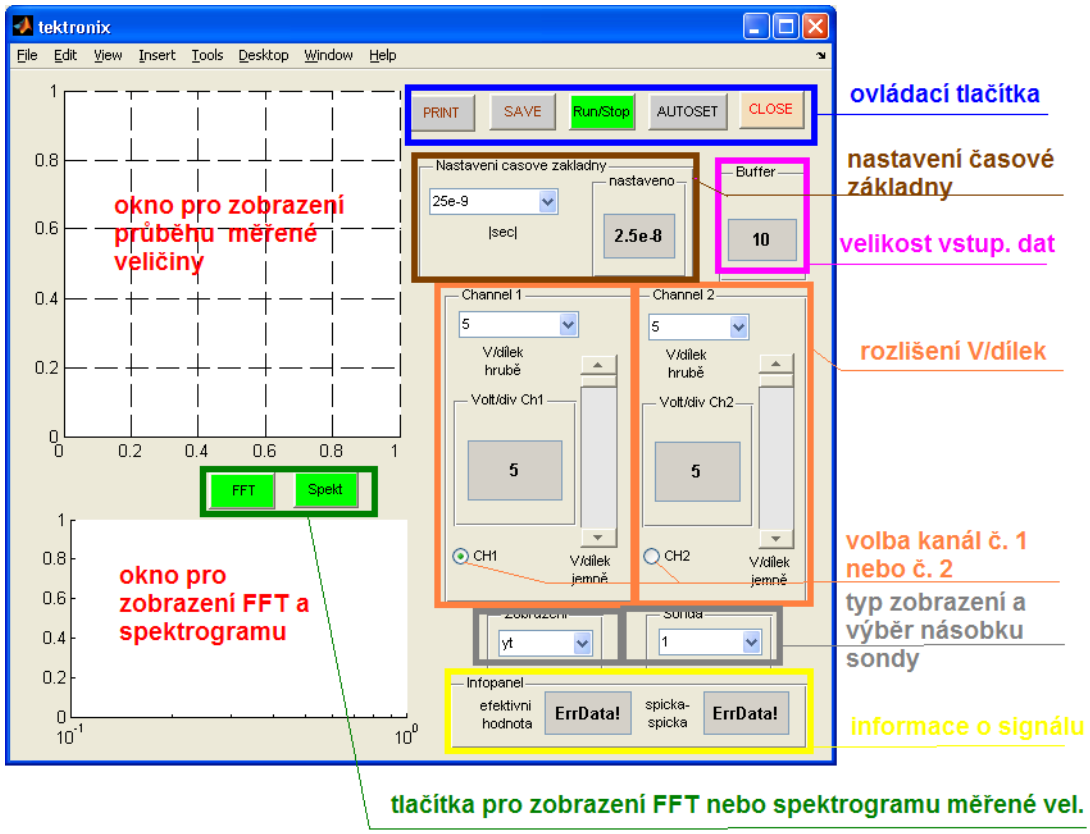


Obr. 8: Pohled na panel GUI

2.2.2 Sestavení jednotlivých funkčních bloků do GUI

Panel GUI osciloskopu Tektronix na obrázku č. 9 je tvořen z jednotlivých grafických prvků, které do panelu GUI přiřazujeme podle způsobu jejich použití (s ohledem jejich předdefinované funkce v programu MATLAB nebo podle jejich využitelnosti pro ovládání osciloskopu). V programu jsem využil **dvě pole axes** pro zobrazení grafů (naměřený průběh veličiny a FTT nebo spektrogram), **7 tlačítek** (pro tisk průběhu signálu, uložení naměřených dat, pro zapnutí/vypnutí aktuálníhoho zobrazení měřené veličiny, tlačítko pro automatické nastavení osciloskopu pro měřený signál a tlačítko pro ukončení programu – uzavření GUI, tlačítko pro zobrazení/vymazání Fourierovy transformace a obdobné tlačítko pro spektrogram). Dále jsem v GUI využil **6 editačních políček** (nastavení nebo kontrola hodnoty časové základny, definice velikosti počtu řádků matice vstupních dat, nastavení zobrazení V/dílek pro kanál č. 1 nebo č. 2 a taktéž dvě informativní tlačítka pro zobrazení efektivní hodnoty měřeného signálu a pro zobrazení maximálního rozdílu vstupních dat – rozdíl mezi špičkami), **5 výběrových menu** (výběr časové základny, nastavení V/dílek pro oba kanály, nastavení hodnoty sondy a nastavení

formátu zobrazení) a také **dva posuvníky** pro jemné nastavování rozlišení V/dílek při zobrazení průběhu.



Obr. 9: Bloky a tlačítka panelu GUI

Z hlediska ovládání funkce (nastavení) osciloskopu spolu tvoří funkční celky nebo se vzájemně doplňují následující komponenty:

- výběrové menu a editační políčko v sekci nastavení časové základny,

- výběrové menu, posuvník a editační pole v komponentě volba kanálu č. l nebo kanálu č. 2,

- zaškrtávací políčka pro volbu kanálu č. 1 a kanálu č. 2,

- tlačítka FFT a Spekt,

Více o funkci jednotlivých tlačítek a vazbách mezi nimi bude popsáno v kapitolách 2.2.3 a 2.2.4 této práce.

2.2.3 Popis funkce GUI

Po spuštění grafického uživatelského rozhraní (zapsáním povelu `tektronix` v příkazovém řádku programu MATLAB nebo dvojklikem myší na název programu tektronix.m) dojde k inicializaci spojení počítače s osciloskopem – je definován způsob spojení počítače a osciloskopu (přes rozhraní USB) a v případě, že je přístroj fyzicky připojen, dojde k propojení osciloskopu s počítačem.

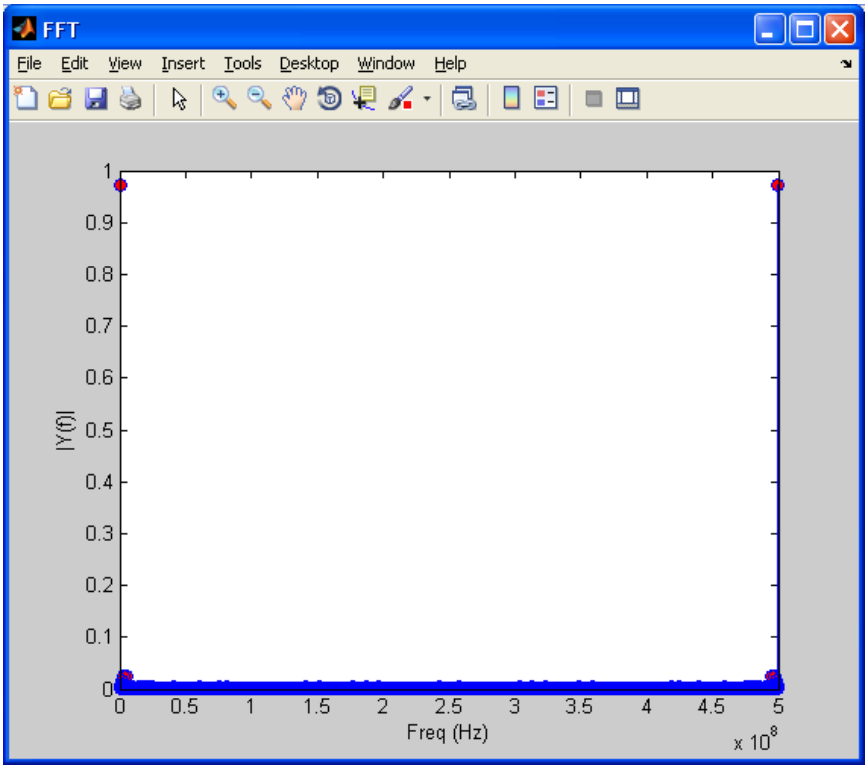
V dalším kroku jsou nastaveny počáteční podmínky, mimo jiné je nastaven počet řádků matice dat (proměnná `handles.buffer`) a definovány matice (`handles.X`, `handles.Y` a `handles.fft_data`), do kterých budou načítána vstupní data a výsledek FFT. Je definována vzorkovací frekvence osciloskopu a Nyquistova frekvence (`handles.f_vz` a `handles.f_N`), jsou vytvořeny pomocné proměnné, které využijeme k testování stisknutí tlačítka „Run/Stop“ (`handles.f`) a k testování, zda byla načtena vstupní data (`handles.probehlo_cteni_dat`).

Dále je provedeno nastavení kanálu, ze kterého budeme číst data, v našem případě z kanálu č. 1, nastavení hodnoty časové základny na výchozí hodnotu 25 ns, nastavení hodnoty rozlišení zobrazení na 5 V/dílek při hodnotě přepínače sondy 1x. Výchozí nastavení barvy tlačítek „Run/Stop“, „FFT“ a „Spekt“ je zelená, po stisku libovolného z těchto tlačítek bude jeho barva změněna na modrou.

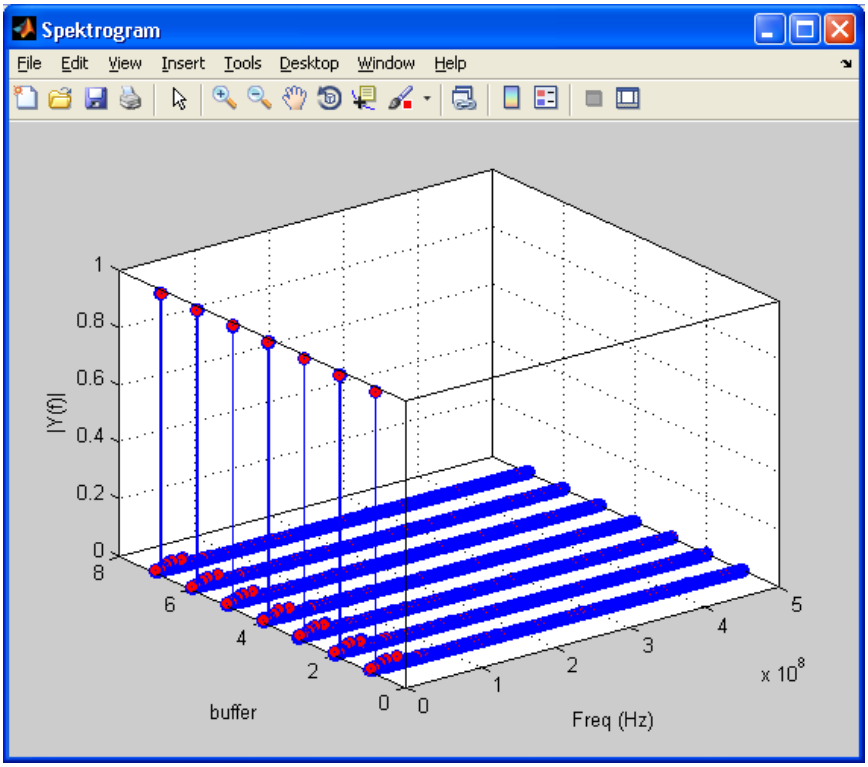
Grafické rozhraní nyní čeká na další zásah uživatele. Pokud chceme zobrazit průběh měřené veličiny, můžeme to udělat dvěma způsoby. Můžeme tak učinit stiskem tlačítka „Autoset“ (dojde k automatickému nastavení parametrů osciloskopu, k jednorázovému načtení dat a jejich vykreslení při optimalizaci pro zobrazení velličiny) nebo stiskem tlačítka „Run/Stop“. Po stlačení tlačítka „Run/Stop“ dojde k průběžnému plnění matice vstupních dat měřenými daty a jejich průběžnému vykreslováni na obrazovku. Po zaplnění všech řádků matice dat (počet řádků je definován pomocnou proměnnou `buffer`) dojde k průběžnému přepisování nejstarších dat daty novými. Přepisování naměřených dat na jedné straně omezuje délku zaznamenaného průběhu měřené veličiny, na druhé straně však zabrání „zamrznutí“ počítače z důvodu zahlcení operační paměti u počítačů s nedostatečným hardwarem (malá operační paměť, malý výkon procesoru). Ukončení vykreslování snímané (měřené) veličiny provedeme opětovným stiskem tlačítka „Run/Stop“, přičemž v okně GUI zůstanou vykreslena poslední naměřená data a jim odpovídající průběh formou statického obrázku vstupních dat.

V průběhu zobrazování měřené veličiny (a nejen při něm) je možno měnit základní nastavení osciloskopu – hodnotu časové základny, hodnotu rozlišení zobrazení (V/dílek), hodnotu násobku měřící sondy, výpočítat a zobrazit průběh FFT apod. Pokud tlačítko „Run/Stop“ není stisknuto, je po každé změně nastavení osciloskopu překreslen poslední zobrazený průběh na průběh s novými parametry.

Z načtených dat je současně s vykreslováním počítána FFT a je ukládána do samostatné proměnné. Fourierovu transformaci nebo spektrogramu signálu je potom možno vykreslit do samostatného okna stiskem tlačítka „FFT“, respektive stiskem tlačítka „Spekt“. Po opětovném stisku příslušného tlačítka dojde k vymazání zobrazeného průběhu. S ohledem na skutečnost, že vykreslovaný průběh FFT a spektrogram se zobrazuje v relativně malém okně, může uživatel po stisku tlačítka myši na příslušný průběh tento otevřít v samostatném okně (figure FFT nebo figure Spektrogram – viz obrázek č. 10 a 11) a dále pak s tímto oknem (vykresleným průběhem) pracovat.



Obr. 10: Okno průběhu FFT



Obr. 11: Okno Spektrogram

Naměřená data je možno z grafického rozhraní exportovat. Data tak můžeme uložit do výstupního souboru s požadovaným jménem (data jsou ukládána ve formátu ascii a jsou uložena vždy do jednoho společného souboru v aktuálním pracovním adresáři, následně je data možno uložit samostatně pro jejich x-ovou a y-ovou složku) nebo jejich průběh můžeme vytisknout (respektive uložit jejich grafickou podobu). Výchozím formátem pro tisk z GUI je formát obrázku (*.jpeg). Ukončení práce s GUI je možno provést stiskem tlačítka „Close“, po němž dojde uzavření všech aktivních oken.

Podrobný funkční popis jednotlivých ovládacích tlačítek, jejich zpětných vazeb bude popsán v následující kapitole.

2.2.4 Detailní popis funkčních celků programu

Vytvoření a otevření GUI:

```
function varargout = tektronix(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
'gui_Singleton', gui_Singleton, ...
```

```
'gui_OpeningFcn', @tektronix_OpeningFcn, ...
'gui_OutputFcn', @tektronix_OutputFcn, ...
'gui_LayoutFcn', [] , ...
'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
function tektronix_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
```

Slouží k definování funkce `tektronix`. Do funkce vstupuje proměnlivé množství vstupních argumentů, které jsou pomocí příkazu `varargin` uloženy do polí buněk (cell array) a zároveň není dopředu znám počet výstupních proměnných pro vykonání funkce. Tyto proměnné jsou obdobně jako vstupní parametry uloženy do proměnných typu buňka (cell).

```
!C:\VXIpnp\WINNT\TekVISA\Bin\TekInstrMgr.exe &
```

Příkaz zajistí spuštění VISA softwaru v počítači. Software VISA je nezbytný pro zajištění komunikace PC s osciloskopem přes rozhraní USB.

```
interfaceObj=instrfind('Type','visa-usb','RsrcName',
'USB0::0x0699::0x0364::C053721::0::INSTR','Tag','');
```

Příkazem dojde k vyhledání a přiřazení „fyzického“ rozhraní USB (do příkazu je nutno zadat jeho plné jméno) k rozhraní osciloskopu Tektronix (definovaný jako objekt `Obj`).

```
if isempty(interfaceObj)
interfaceObj = visa('TEK','USB0::0x0699::0x0364::C053721::0::INSTR');
else
fclose(interfaceObj);
interfaceObj = interfaceObj(1);
end
```

Sekvencí příkazů dojde k přiřazení rozhraní. Pokud není rozhraní standardu VISA, název přístroje `Tek` a název rozhraní USB `USB0::0x0699::0x0364::C053721::0::INSTR` (viz obrázek č. 3 na straně 14) dojde k jeho uzavření.

```
handles.deviceObj = icdevice('tektronix_tds2002.mdd', interfaceObj);
connect(handles.deviceObj);
```

Zde je definován ovladač pro osciloskop (zpracovaný firmou MathWorks pro příslušný typ osciloskopu), v následujícím kroku dochází k propojení Tektronixu s počítačem (programem MATLAB).

```
handles.X = []; %Xová data
handles.Y = []; %Yová data
handles.fft_data= []; %Fourier data
```

Definování proměnné `handles.X`, `handles.Y` a `handles.fft_data` – jedná se o proměnné typu matice, do kterých jsou v dalším běhu programu ukládána data a dle potřeby čtena a zobrazována. Do matic jsou tedy uložena data popisující měřenou veličinu (vstupní data, jejich x-ovou a y-ovou složku) nebo výsledek FFT.

```
handles.f_vz = 1e+9; %f_vz - sample Tektronix (1Giga)
handles.f_N = handles.f_vz / 2; %Nyquist kmitocet
```

Definování proměnné `handles.f_vz` (samplovací kmitočet Tektronixu je 1 GS/s), `handles.f_N` (Nyquistův[2] kmitočet), tyto proměnné jsou v průběhu programu využity pro vykreslení FFT nebo pro zobrazení spektrogramu.

```
handles.c = 0; %tlačítko Run/Stop není stlačeno
```

Pomocná proměnná, která v programu slouží k jednoznačné identifikaci, zda je tlačítko „Run/Stop“ stisknuto (hodnota proměnné je po stisku tlačítka rovna 1), výchozí hodnota proměnné je 0, což odpovídá skutečnosti, že tlačítko není stlačeno.

[2] Nejvyšší kmitočet ve vzorkované veličině může být roven maximálně polovině vzorkovací frekvence. Tento nejvyšší kmitočet je nazýván Nyquistovou frekvencí.

```
handles.buffer=10; %defaultní nastavení počtu řádků v matici pro uložení dat
```

Pomocná proměnná, určuje počet řádků matice vstupních dat (`handles.X` a `handles.Y`), které budou využity pro ukládání dat při komunikaci počítače s osciloskopem. Pokud je použita příliš velká hodnota, může dojít při čtení a ukládání dat k zahlcení operační paměti počítače. Prvotní nastavení proměnné je z tohoto důvodu definováno na hodnotu 10 (matice o rozměru 10 řádků x 2 500 sloupců).

```
handles.probehlo_cteni_dat = 0; %pomocna promenna - test cteni dat
```

Pomocná proměnná, v programu je využita při testování, zda již došlo k přenosu dat z osciloskopu do počítače. V případě, že se program chce odvolat na neexistující data (nedošlo k přenosu dat do počítače), bude zobrazeno chybové hlášení (funkce `pomoc`).

```
handles.popup=[25e-9,50e-9,100e-9,250e-9,500e-6,1e-6,2.5e-6,5e-6,10e-6,…
25e-6,50e-6,100e-6,250e-6,500e-6,1e-3,2.5e-3,5e-3,10e-3,25e-3,50e-3,…
100e-3,250e-3,500e-3,1,2.5,5,10,25,50]; %definování vektoru cas. zakladny
```

Pomocná proměnná (matice, ve které jsou uloženy všechny možnosti nastavení časové základny osciloskopu), v programu je využita k propojení výběrového menu (přednastavené hodnoty pro výběr časové základny osciloskopu) s editačním polem časové základny, které umožňuje přímé zadání hodnoty časové základny (bez znalosti předdefinovaných hodnot).

```
set(handles.efektivni,'String','ErrData!');
set(handles.spicka_spicka,'String','ErrData!');
```

Do informačního pole pro zobrazení velikosti efektivní hodnoty měřené veličiny a rozdílu mezi maximem a minimem měřeného průběhu (špička–špička) je při počáteční inicializaci GUI načten řetězec „ErrData!“. Tento řetězec by měl uživatele upozornit na skutečnost, že ještě nedošlo k načtení dat. Při jakémkoli načtení dat z osciloskopu je řetězec nahrazen skutečně vypočtenými hodnotami efektivní hodnoty a rozdílem špička–špička měřeného průběhu.

```
set(handles.edit_buffer,'String',num2str(handles.buffer)); %Zápis do Buffer
```

zápis hodnoty, uložené v proměnné `handles.bufer` do editačního pole přepisovatelného textového pole „Buffer“. Příkazem `num2str` dojde k převodu číselné hodnoty proměnné

`handles.buffer` na řetězec, příkazem `set` dojde k přiřazení požadované vlastnosti grafického objektu (v našem případě zápis čísla 10 do editačního pole proměnné Buffer).

```
set(handles.deviceObj.Channel(1), 'Probe',1); %základní nastavení sondy 1x
```

Defaultní nastavení hodnoty přepínače měřící sondy na hodnotu násobku 1 (vlastnost osciloskopu `Probe`).

```
set(handles.volt_dilek_ch1, 'String', '5'); %nastavení V/dílek pro ch1
set(handles.volt_dilek_ch2, 'String', '5'); %nastavení V/dílek pro ch2
```

Zápis hodnoty 5 Volt/dílek do editačních polí příslušných kanálů. Příkazem `set` dojde k přiřazení požadované vlastnosti grafického objektu typu řetězec (zápis čísla 5 do příslušného editačního pole kanálu je v textovém formátu).

```
set(handles.deviceObj.Measurement(1),'Source','channel1');  %nastavení kanálu č.1 jako počáteční stav
```

Nastavení zdroje dat (vlastnosti objektu `Source`), ze kterého bude zařízení číst, na hodnotu `channel1` (čtení dat bude prováděno z kanálu č. 1 osciloskopu)

```
handles.kan=get(handles.deviceObj.Measurement(1),'Source'); %načtení kanálu do proměnné handles.kan
```

Do proměnné `handles.kan` je uloženo, ze kterého zdroje (kanálu) právě probíhá čtení dat vstupní proměnné

```
set(handles.edit_casova_zakladna,'String','2.5e-8'); %zápis hodnoty 2,5 e-8 do edit pole
```

Prvotní nastavení editačního pole pro nastavení časové základny na hodnotu 250 µV. Hodnota editačního pole je provázána s výběrovým menu pro časovou základnu, takže hodnota 250 µV odpovídá rovněž nastavení hodnoty časové základny osciloskopu.

```
set(handles.deviceObj.Channel(1), 'Scale',5);
set(handles.volt_dilek_jemne_ch1, 'Value',1);

set(handles.deviceObj.Channel(2), 'Scale',5);
set(handles.volt_dilek_jemne_ch2, 'Value',1);
```

Touto sekvencí příkazů dojde k nastavení hodnoty zobrazení osciloskopu na hodnotu 5 V/dílek (vlastnost `Scale`). Dále je příkazem `set` nastavena výchozí poloha posuvníku (`handles.volt_dilek_jemne_ch1` a její vlastnost `Scale`. Hodnota 1 odpovídá horní poloze posuvníku). Postup nastavení kanálu č. 2 je totožný s postupem pro kanál č. 1 (jedná se o identickou sekvenci příkazů s parametrem čísla kanálu 2).

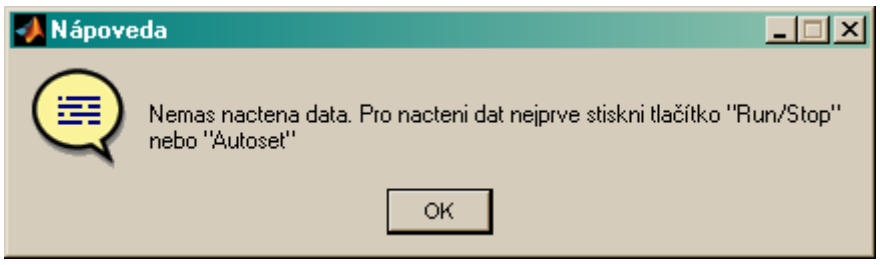
```
guidata(hObject, handles);
```

Příkazem `guidata` dojde k uložení grafického rozhraní

Funkce pomocné

```
function pomoc(hObject, eventdata, handles)
helpdlg ('Nemas nactena data. Pro nacteni dat nejprve stiskni tlačítko "Run/Stop" nebo "Autoset"','Nápoveda');
```

Funkcí `pomoc` je definováno zobrazení hlášení v provedení help okna (vzhled okna je šablonou programu MATLAB). Zobrazí se text `Nemas nactena data. Pro nacteni dat nejprve stiskni tlačítko "Run/Stop" nebo "Autoset"` (viz obrázek č. 12), v horní části okna (horní liště) je zobrazen text `Nápoveda`. Funkce `pomoc` je aktivována pokaždé, jestliže dojde k požadavku uživatele na zpracování neexistujících dat programem (nedošlo k přenesení dat z osciloskopu do PC).



Obrázek č. 12: Okno Nápovědy – pokyn pro načtení dat

```
function info(hObject, eventdata, handles)
if max(max(handles.X))~=0 %jestli jsou nactena data
rozdil=max(max(handles.Y))-min(min(handles.Y)); %rozdil spicka - spicka
set(handles.spicka_spicka, 'String',num2str(rozdil)); else
set(handles.spicka_spicka, 'String','ErrData!'); %nejsou nactena data
end
if max(max(handles.X))~=0
t=max(size(handles.Y(end,:))); %zjisteni poctu namerenych hodnot
z=(handles.Y(end,:))*(handles.Y(end,:))'; %vypocet mocniny do pomocne promenne
e=1/t*z; %dilci vypocet ef. hodnoty
ef=sum(e(end,:)); %dilci vypocet ef. hodnoty
si_ef=sqrt(ef); %vypoctena efektivni promenna
set(handles.efektivni, 'String',num2str(si_ef)); %nastaveni efektivni hodnoty
do pole efektivni
else
set(handles.efektivni, 'String','ErrData!'); %nejsou nactena data
end
```

Funkcí `info` je definován zápis hodnoty do editačních polí v Infopanelu, odpovídajících efektivní hodnotě měřené veličiny a rozdílu špička–špička měřené veličiny (proměnné `handles.spicka_spicka` a `handles.efektivni`). Poté, co jsou z osciloskopu načtena data (maximální hodnota matice vstupních dat je nenulová) je spočítán rozdíl mezi maximální a minimální hodnotu měřené veličiny, tato hodnota je uložena do proměnné `rozdil`, proměnná je po převedení na řetězec příkazem `set` uložena do proměnné `handles.spicka_spicka` (její vlastnosti `String`).

Do proměnné `handles.efektivni` je v případě, že jsou vstupní data načtena, uložena efektivní hodnota měřeného průběhu. Tato hodnota je spočítána pomocí obecného vzorce pro výpočet efektivní hodnoty a do editačního pole zapsána příkazem `set`, vlastnost proměnné `handles.efektivni String`.

Pokud nejsou do PC data načtena (test se provádí jednou pro obě proměnné – efektivní hodnotu i rozdíl špička-špička), jsou do obou proměnných uloženy řetězce „ErrData!“ a ty následně zobrazeny v editačních polích GUI – viz obr. 13.

Obrázek č. 13: Pole Infopanelu – dosud nejsou načtena data

```
function ukaz_FFT(hObject, eventdata, handles)%disp('zobraz_FFT')
invoke(handles.deviceObj,'autoset'); %autoset
set(handles.edit_casova_zakladna,'String',…
num2str(get(handles.deviceObj.Acquisition(1), 'Timebase')));
caszakl=str2double(get(handles.edit_casova_zakladna, 'String'));
vypocet = (1/caszakl)*handles.popup; %do pozice popup menu, kterou hledam
dostanu vypoctem 1
index=findstr(vypocet,1); %do index je nactena hledana pozice (cislo 1)
set(handles.vyber_casova_zakladna, 'Value',index); %aktualizace popup menu
if handles.kan == 'channel1'
set(handles.volt_dilek_ch1,'String',…
num2str(get(handles.deviceObj.Channel(1), 'Scale')));
retez1 = get(handles.volt_dilek_hrube_ch1, 'String');
hodn1 = str2double(retez1{get(handles.volt_dilek_hrube_ch1, 'Value')});
val1=str2double(get(handles.volt_dilek_ch1,'String'))/hodn1;
set(handles.volt_dilek_jemne_ch1,'Value',val1);
else
set(handles.volt_dilek_ch2,'String',…
num2str(get(handles.deviceObj.Channel(2), 'Scale')));
retez2 = get(handles.volt_dilek_hrube_ch2, 'String');
hodn2 = str2double(retez2{get(handles.volt_dilek_hrube_ch2, 'Value')});
val2=str2double(get(handles.volt_dilek_ch2,'String'))/hodn2;
set(handles.volt_dilek_jemne_ch2,'Value',val2);
end
N = length(handles.fft_data(end,:));%pocet vzorku
Spektrum_dvoustranne = (handles.fft_data(end,:))/N;
delta_f = handles.f_N / N;
f_jednostranne = (0 : delta_f : handles.f_N-delta_f); %osa Xová (frekvence)
if mod(N, 2) == 1 %Lichy pocet vzorku - prevod obustranneho spektra na
jednostrane
handles.Amplitudove_Spektrum_jednostranne = [abs(Spektrum_dvoustranne(1)),…
2*abs(Spektrum_dvoustranne(2 : N-1 ))];
else %sudy pocet vzorku
handles.Amplitudove_Spektrum_jednostranne = [abs(Spektrum_dvoustranne(1)),…
2*abs(Spektrum_dvoustranne(2 : N))];
end
axes(handles.zobraz_FFT);
fft=stem(f_jednostranne, handles.Amplitudove_Spektrum_jednostranne);
```

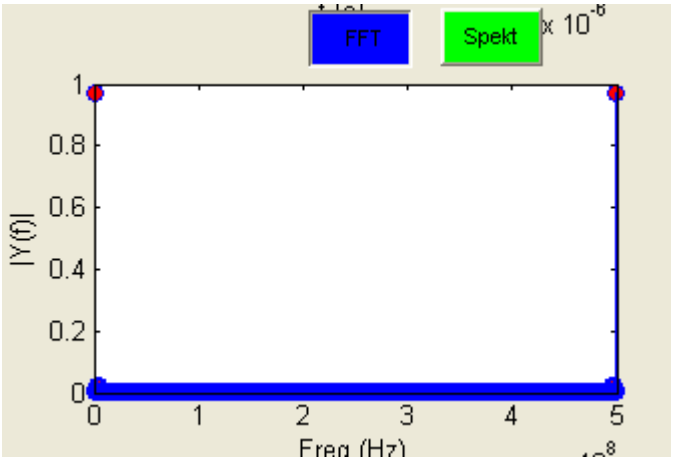
```
maxy=ceil(max(handles.Amplitudove_Spektrum_jednostranne)); %zaokrouhleni
maxima osy y na hodnotu nahoru
ylim([0 maxy]); %nastaveni hranice osy y
set(fft,'MarkerFaceColor','red'); %nastaveni bodu grafu (cervena)
xlabel('Freq (Hz)')%nazev osy x
ylabel('|Y(f)|') %nazev osy y
set(handles.zobraz_FFT,'NextPlot','replacechildren','ButtonDownFcn',…
@(hObject, eventdata)tektronix('zobraz_FFT_ButtonDownFcn', hObject,…
eventdata, guidata(hObject)));
guidata(hObject, handles);
```

Funkce `ukaz_FFT` slouží k zobrazení stonkového grafu Fourierovy transformace měřené veličiny v definovaném okně GUI. Po zavolání této funkce je nejprve proveden autoset osciloskopu. Při funkci autoset je provedena změna nastavení časové základny osciloskopu a optimalizace zobrazení (V/dílek). Z toho důvodu je do editačního pole časové základny proveden zápis nové hodnoty časové základny osciloskopu, proveden výpočet pozice hodnoty základny v matici výběrového menu a následně vybrána a zobrazena hodnota skutečné velikosti časové základny ve výběrovém menu (hodnota výběrového menu a editačního pole musí být stejná).

V dalším kroku je zjištěn zdroj signálu (čtení z kanálu č. 1 nebo kanálu č. 2) a podle vstupu je provedena aktualizace odpovídajícího editačního pole nastavení V/dílek a příslušného posuvníku (pro kanál č. 1 nebo kanál č. 2) s nastavením osciloskopu.

Do proměnné `N` je uložen počet vzorků měřené veličiny (příkazem `length` je zjištěn počet prvků matice `handles.fft_data`, respektive jejího posledního řádku dat `(end,:)`), do proměnné `delta_f` je vypočítán a uložen krok pro zobrazení osy x grafu a do proměnné `f_jednostranne` jsou uložena data popisující xovou osu (frekvence). Do proměnné `Spektrum_dvoustranne` je uložen výsledek FFT měřené veličiny (FFT posledního řádku dat matice `handles.fft_data`. V následném kroku je zjišťováno, zda je počet vzorků sudé nebo liché číslo (zda je zbytek po dělení modulo 2 roven 0 nebo 1) a podle počtu vzorků je z oboustranného spektra vytvořeno jednostranné spektrum (jednostranné spektrum má širší využitelnost v elektrotechnice). Jednostranné spektrum je definováno jako absolutní hodnota dvojnásobku hodnoty dvojstranného spektra. Pro lichý počet prvků je poslední prvek matice dat FFT definován jako předposlední vzorek měřeného průběhu. Následně je do proměnné `fft` uložen stonkový graf s prvky odpovídajícími jednostrannému průběhu FFT, tento je zobrazen v okně GUI (průběh je zobrazen například na obrázku č. 14).

Pro korektní popis os je nutno přepočítat hodnoty, které budou v grafu zobrazovány - krok mezi jednotlivými vzorky x-ové osy je `delta_f`, x-ová osa (je popsána jako `f_jednostranne`) je v rozmezí 0 až Nyquistův kmitočet, data y-ové osy jsou uložena v proměnné `handles.Amplitudove_Spektrum_jednostrane`. Do proměnné `maxy` je uložena maximální hodnota FFT měřeného průběhu, příkazem `ylim([0 maxy])` jsou definovány krajní hodnoty zobrazení na ose y, příkazem `set(fft,'MarkerFaceColor','red')` je definována červená barva koncových bodů stonkového grafu. Přikazem `xlabel` a `ylabel` je definován popis jednotlivých os. Předposlední příkaz funkce, začínající sekvencí `set(handles.zobraz_FFT,@(hObject, eventdata)tektronix…` definuje chování programu po kliknutí myši na okno grafu FFT. Po detekování stisku myši je zavolána funkce `zobraz_FFT_ButtonDownFcn`, která v samostatném okně zobrazí graf Fourierovy transformace. Příkazem `guidata` dojde k uložení celého GUI.



Obrázek č. 14: Okno s průběhem FFT

```
function zobraz_spektrum(hObject, eventdata, handles)
invoke(handles.deviceObj, 'autoset'); %autoset
set(handles.edit_casova_zakladna, 'String',…
num2str(get(handles.deviceObj.Acquisition(1), 'Timebase')));
caszakl=str2double(get(handles.edit_casova_zakladna, 'String'));
vypocet = (1/caszakl)*handles.popup; %do pozice popup menu, kterou hledam,
dostanu vypoctem 1
index=findstr (vypocet,1); %do promenne index je nactena hledana pozice (cislo
1)
set(handles.vyber_casova_zakladna, 'Value',index); %aktualizace popup menu
if handles.kan == 'channel1'
```

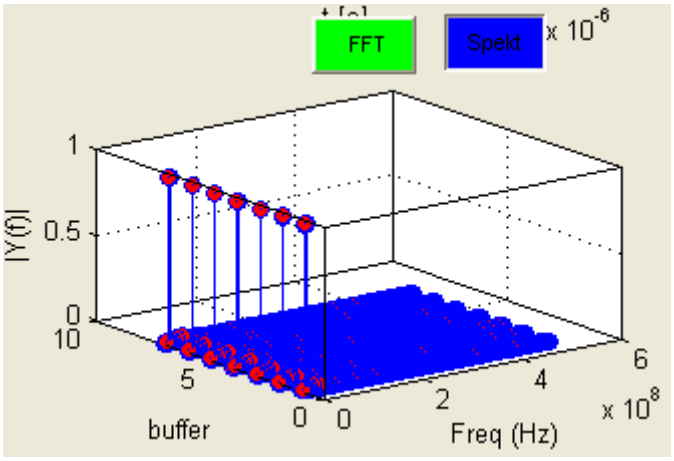
```
set(handles.volt_dilek_ch1,'String',…
num2str(get(handles.deviceObj.Channel(1), 'Scale')));
retez1 = get(handles.volt_dilek_hrube_ch1, 'String');
hodn1=str2double(retez1{get(handles.volt_dilek_hrube_ch1, 'Value')});
val1=str2double(get(handles.volt_dilek_ch1,'String'))/hodn1;
set(handles.volt_dilek_jemne_ch1,'Value',val1);
else
set(handles.volt_dilek_ch2,'String',…
num2str(get(handles.deviceObj.Channel(2), 'Scale')));
retez2 = get(handles.volt_dilek_hrube_ch2, 'String');
hodn2 = str2double(retez2{get(handles.volt_dilek_hrube_ch2,'Value')});
val2=str2double(get(handles.volt_dilek_ch2,'String'))/hodn2;
set(handles.volt_dilek_jemne_ch2,'Value',val2);
end
Buf=size(handles.fft_data, 1); %pocet nactenych radku
N = length(handles.fft_data); %pocet vzorku
delta_f = handles.f_N / N; %krok na ose frekvence
[osa_X, Y_f] = meshgrid (0:delta_f:handles.f_N-delta_f,…
1:size(handles.fft_data, 1));
for k = 1 : Buf
if mod(N, 2) == 1 %Lichy pocet vzorku - jednostranne spektrum
vektor_fft(k,:) = [abs(handles.fft_data(1))/N,…
2*abs(handles.fft_data(2 : N-1))/N];
else
vektor_fft(k,:) = [abs(handles.fft_data(1))/N,…
2*abs(handles.fft_data(2:N))/N];
end
end
axes(handles.zobraz_FFT);
spek=stem3(osa_X, Y_f, vektor_fft); %graf spektra
ylim([0 10]);
set(spek,'MarkerFaceColor','red'); %nastaveni bodu spektra (cervena)
xlabel('Freq (Hz)') %nazev x-ove osy
ylabel('buffer') %nazev y-ove osy
zlabel('|Y(f)|') %nazev z-osy
set(handles.zobraz_FFT,'NextPlot','replacechildren','ButtonDownFcn',…
@(hObject, eventdata)tektronix('zobraz_FFT_ButtonDownFcn',…
hObject, eventdata, guidata (hObject)));
guidata(hObject, handles);
```

Funkce `zobraz_spektrum` slouží k zobrazení 3-D stonkového grafu průběhu spektra (spektrogramu) měřené veličiny v okně GUI. Po zavolání této funkce je nejprve proveden autoset osciloskopu. Při funkci autoset je provedena změna nastavení časové základny osciloskopu, optimalizace zobrazení V/dílek. Z toho důvodu je do editačního pole časové základny proveden zápis nové hodnoty časové základny osciloskopu, proveden výpočet pozice hodnoty základny v matici výběrového menu, vybrána jeho pozice a ve výběrovém menu zobrazena hodnota skutečné hodnoty časové základny (hodnota výběrového menu a editačního pole musí být stejná).

V dalším kroku je zjištěn zdroj signálu (čtení z kanálu č. 1 nebo kanálu č. 2) a podle vstupu je provedena aktualizace hodnoty odpovídajícího editačního pole (nastavení V/dílek) a příslušného posuvníku (pro kanál č. 1 nebo kanál č. 2) s nastavením osciloskopu.

Do proměnné `Buf` je příkazem `size` načten počet řádků matice dat (`handles.fft_data`). Do proměnné `N` je pomocí příkazu `length` zjištěn počet prvků (délka matice `handles.fft_data`, jejího posledního řádku dat `(end,:))`, což odpovídá počtu vzorků měřené veličiny, do proměnné `delta_f` je vypočítán a uložen krok pro zobrazování osy x grafu. Pro vykreslení průběhu spektra je nutno zobrazit 3-D graf, z tohoto důvodu je nutno definovat z- osu (FFT) v závislosti na osách x a y (frekvence a počet zaplněných řádků matice dat FFT - odpovídá proměnné `Buf`). V následném kroku je zjišťováno, zda je počet vzorků sudé nebo liché číslo (zda je zbytek po dělení modulo 2 roven 0 nebo 1) a podle počtu vzorků je z oboustranného spektra vytvořeno jednostranné spektrum (jeho absolutní hodnota, pro lichý počet prvků odpovídá poslední prvek matice spektra předposlednímu vzorku). Následně je do proměnné `spekt` uložen trojrozměrný stonkový graf s prvky odpovídajícími několika po sobě jdoucích jednostranných průběhů FFT, tento je rovněž zobrazen v okně v GUI (viz obrázek č. 15).

Příkazem `ylim([0 10])` jsou definovány mezní hodnoty zobrazení v ose y (počet načtených průběhů, což odpovídá počtu řádků matice dat FFT), příkazem `set(spekt,'MarkerFaceColor','red')` je definována červená barva koncových bodů 3-D stonkového grafu. Přikazy `xlabel`, `ylabel` a `zlabel` jsou definovány popisky os. Příkazem začínající sekvencí `set(handles.zobraz_FFT,…@(hObject, eventdata)tektronix…` je definováno chování programu po kliknutí myši na okno spektrogramu. Je zavolána funkce `zobraz_FFT_ButtonDownFcn`, která zobrazí v samostatném okně graf průběhu spektra. Příkazem `guidata` dojde k uložení GUI.



Obrázek č. 15: Okno spektrogramu

```
function [X, Y, fft_data] = jednorazove_nacteni(hObject, eventdata,
handles)

groupObj = get(handles.deviceObj, 'Waveform'); %"zadost" o signal

groupObj = groupObj(1);

handles.probehlo_cteni_dat =handles.probehlo_cteni_dat + 1; %budu cist
data, proto pomocna promenna neni nulova

if handles.kan == 'channel1'

[Y, X, Yu, Xu] = invoke(groupObj, 'readwaveform', 'channel1');
%nacteni z kanalu c.1

else

[Y, X, Yu, Xu] = invoke(groupObj, 'readwaveform', 'channel2');
%nacteni z kanalu c.2

end

fft_data = fft(Y); %do promenne je vypoctena FFT
```

Funkcí `[X,Y,fft_data] = jednorazove_nacteni` je definováno jednorázové načtení dat (připojení osciloskopu, načtení dat do matice dat [Y, X, fft_data]). Nejprve je příkazem `get` „požádáno“ o časový průběh signálu (položka `Waveform`). Následně je inkrementována hodnota proměnné `handles.probehlo_cteni_dat` a její význam se uplatní v dalším průběhu programu (aktuální hodnota je porovnávána s její výchozí hodnotou a při nerovnosti s 0 nedochází k vykreslení okna nápovědy). Po testu, ze kterého kanálu budou data načítána, jsou do matice dat načtena vstupní data – položka `readwaveform` s parametrem `channel1` nebo `channel2`. Výstupem funkce je matice načtených dat, které

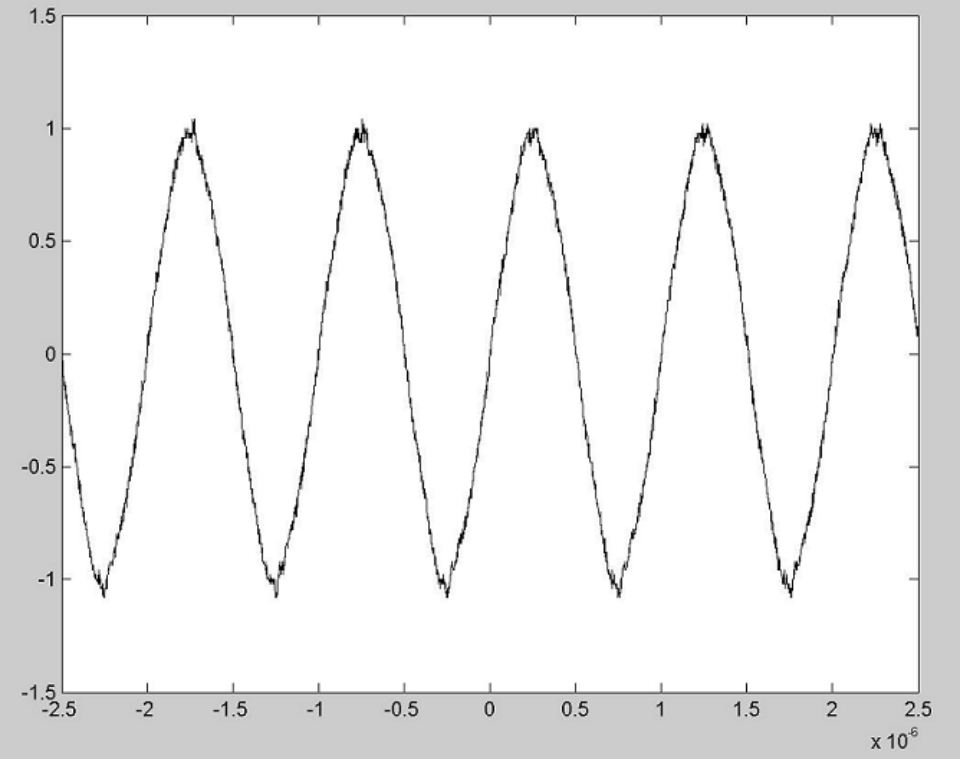
po vykreslení odpovídají 2-D zobrazení průběhu signálu a matice dat odpovídající hodnotě Fourierovy transformace čtených dat.

Funkce volané po stisku tlačítka:

Obrázek č. 16: Tlačítka GUI tektronix

```
function print_Callback(hObject, eventdata, handles)
if handles.probehlo_cteni_dat == 0
pomoc(hObject, eventdata, handles);
pause(2);
end
figure
plot(handles.X(end,:), handles.Y(end,:), 'k');
print ('-f1', '-djpeg', 'vykresleny_prubeh.jpg');
close Figure 1
```

Funkce `print_Callback` nejprve otestuje, zda již v předchozím průběhu programu došlo k čtení dat. Pokud tomu tak není, je zobrazeno okno nápovědy, které v popředí monitoru setrvá po dobu 2 vteřin (po tuto dobu je zároveň přerušen průběh programu). Příkaz `figure` otevře nové okno, do kterého je následně příkazem `plot(handles.X(end,:), handles.Y(end,:),'k')` vykreslen 2-D graf naměřených hodnot. Příkaz `print('-f1','-djpeg','vykresleny_prubeh.jpg')` provede uložení okna ve formátu obrázku (formát *.jpeg), do aktuálního pracovního adresáře programu MATLAB. Ukládané okno je zobrazeno na obrázku č. 17. Název uloženého souboru pro náš konkrétní případ je v programu definován jako `vykresleny_prubeh.jpeg`. Příkazem `close Figure1` okno s vykresleným průběhem uzavřeme.



Obrázek č. 17: Objekt Figure1 uložený jako obrázek - vykresleny_prubeh.jpg

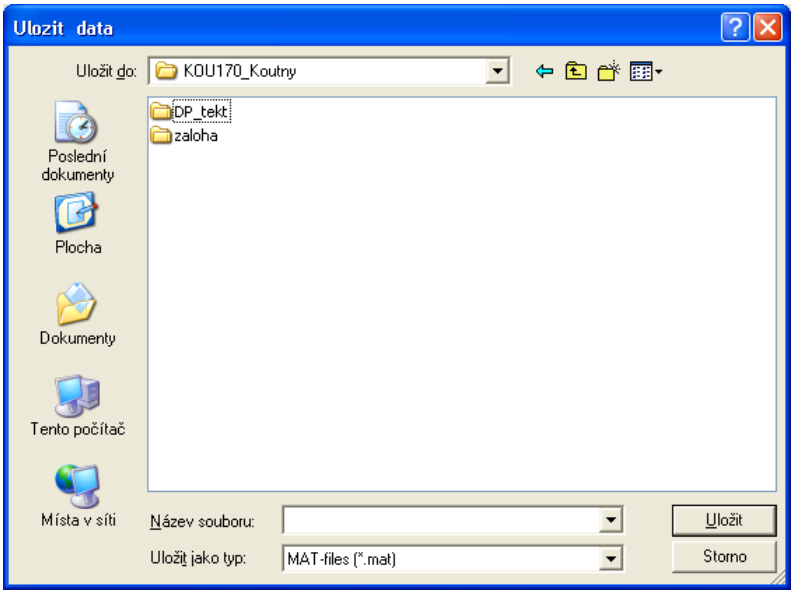
```
function save_Callback(hObject, eventdata, handles)
if handles.probehlo_cteni_dat == 0
pomoc(hObject, eventdata, handles);
pause(5);
end
X = handles.X; %prevzeti dat z promene handle
Y = handles.Y;
save vystup_XY X Y; %ulozeni do vystupniho souboru vystup_XY.mat v pracovnim
adresari
 [jmeno, adresar, format] = uiputfile({'*.mat'; '*.asc'}, 'Ulozit data');
%ukladani X a Y dat samostatne
formaty = {'', '-ascii'};
if jmeno==0 %storno
else
save([adresar, 'X_', jmeno ], 'X', formaty{format});
save([adresar, 'Y_', jmeno], 'Y', formaty{format});
end
```

Při vykonávání funkce `save_Callback` se nejprve otestuje, zda načtení dat již proběhlo. V případě, že komunikace počítače s osciloskopem ještě neproběhla, objeví se v popředí

obrazovky na dobu 5 vteřin aktivní okno `help` (nápovědy), které uživatele upozorní na nutnost načtení vstupních dat.

V případě, že data již načtena jsou, dojde k předání dat z proměnné `handles.X` do proměnné X a `handles.Y`, do proměnné Y. Následným příkazem `save vystup_XY X Y` dojde k uložení naměřených hodnot do jediného souboru se jménem `vystup_XY.mat`, který bude uložen v pracovním adresáři (uložení dat proběhne automaticky bez uživatelského zásahu a možnosti ovlivnit název nebo příponu souboru).

Příkazem `[jmeno,adresar,format]=uiputfile({'*.mat';'*.asc'},'Ulozit data')` dojde k otevření dialogového okna s názvem `Ulozit data`, umožňujícího data uložit v souboru pod libovolným jménem ve formátu dat ascii (příkaz `formaty ={'','-ascii'})`. Následující sekvence příkazů nám umožní data rozdělit na x-ovou a y-ovou část pomocí jediného dialogového okna. Název x-ové části bude `X_libovolný název.mat` a y-ové části analogicky `Y_libovolný název.mat`. V případě volby „Uložit soubor jako“ typ *.asc bude název x-ové části `X_cokoli.asc` a y-ové části bude `Y_cokoli.asc`. Dialogové okno pro uložení dat (rozdělených na dva samostatné soubory) je na obrázku č. 18:



Obrázek č. 18: Dialogové okno pro uložení dat

```
function run_stop_Callback(hObject, eventdata, handles)
handles.probehlo_cteni_dat =1; %budu cist data, pomocna promenna neni nula
handles.c = get(handles.run_stop, 'Value'); %stav tlacítka Run/Stop
while handles.c == 1
set(handles.run_stop, 'BackgroundColor','b'); %barva tlacitka Run/Stop (modra)
- je stlaceno
size(handles.X, 1); %velikost (pocet)rádku
if size (handles.X,1) < handles.buffer
[X, Y, fft_data] = jednorazove_nacteni(hObject, eventdata, handles);
handles.X = [handles.X; X]; %nové data jako poslední
handles.Y = [handles.Y; Y];
handles.fft_data = [handles.fft_data; fft_data];
clear X Y fft_data
else
[X, Y,fft_data] = jednorazove_nacteni(hObject, eventdata, handles);
handles.X = [handles.X(2:end, :); X]; %vypoustí se první rádek
handles.Y = [handles.Y(2:end, :); Y];
handles.fft_data = [handles.fft_data(2:end, :); fft_data];
clear X Y fft_data
end
guidata(hObject, handles);
zobraz_data(hObject, eventdata, handles);
info(hObject, eventdata, handles);
four= get(handles.vykonej_fft, 'Value'); %stav tlacítka FFT
if four == 1
set(handles.vykonej_fft, 'BackgroundColor','b'); %barva tlacitka FFT (modra)
- je stlaceno
set(handles.spektrogram, 'Value',0); %nastav tlacítka spekt nezmácknuto
ukaz_FFT(hObject, eventdata, handles);
four= get(handles.vykonej_fft, 'Value'); %stav tlacítka FFT
else
set(handles.vykonej_fft, 'BackgroundColor','g'); %barva tlacitka FFT (zelene)
- neni stlaceno
end
tl_sp= get(handles.spektrogram, 'Value'); %stav tlacítka spek
if tl_sp==1
set(handles.spektrogram, 'BackgroundColor','b'); %barva tlacitka Spek (modre)
- je stlaceno
set(handles.vykonej_fft, 'Value',0); %nastav tlacítka FFT nezmácknuto
zobraz_spektrum(hObject, eventdata, handles);
```

```
tl_sp= get(handles.spektrogram, 'Value'); %stav tlacítka spek
else
set(handles.spektrogram, 'BackgroundColor','g'); %barva tlacitka Spekt(zelene)
- neni stlaceno
end
handles.c = get(handles.run_stop, 'Value'); %zjistení stavu tlacítka Run/Stop
end
set(handles.vykonej_fft, 'Value',0); %nastav tlacítka FFT nezmácknuto
set(handles.vykonej_fft, 'BackgroundColor','g'); %barva tlacitka FFT (zelene)
- neni stlaceno
set(handles.spektrogram, 'Value',0); %nastav tlacítka spekt nezmácknuto
set(handles.spektrogram, 'BackgroundColor','g'); %barva tlacitka Spekt(zelene)
- neni stlaceno
cla(handles.zobraz_FFT,'reset');
set(handles.run_stop, 'BackgroundColor','g'); %barva tlacitka Run/Stop(zelene)
- neni stlaceno
guidata(hObject, handles);
```

Funkce `run_stop_Callback` zajišťuje průběžné čtení dat z osciloskopu pomocí funkce `jednorazove_nacteni`. Nejprve je do proměnné `handles.probehlo_cteni_dat` uložena hodnota 1 (data jsou zaručeně načtena, nebude nutno zobrazit help okno v dalším průběhu programu). Následně je do proměnné `handles.c` načtena hodnota tlačítka „Run/Stop“ (`handles.run_stop` a její vlastnost `Value`). Pokud je tlačítko stlačené (hodnota `handles.c = 1`) je příkazem `set(handles.run_stop,… 'BackgroundColor', 'b')` nastavena barva tlačítka „Run/Stop“ na modrou, dále je zjišťována velikost proměnné `handles.X` příkazem `size`. Pokud je velikost (počet zaplněných řádků matice naměřených dat) menší než hodnota pomocné proměnná `handles.buffer`, je opětovně zavolána funkce pro jednorázové načtení dat a do matice načtených dat jsou zapisována data a hodnota FFT průběhu (každý příkaz pro načtení reprezentuje jeden řádek matice dat, nově příchozí data jsou řazena jako poslední – příkaz `handles.X = [handles.X; X], handles.Y = [handles.Y;Y]` a `handles.fft_data =[handles.fft_data; fft_data])`.

I po zaplnění předdefinované velikosti matice vstupních dat je nutno zajistit průběžné ukládání nově příchozích dat. Ukládání nových dat je vyřešeno tak, že je nejprve vypuštěn první řádek matice dat (jsou v něm uložena nejstarší data), následně příkazem `handles.X=…`

`[handles.X(2:end,:);X]`, jsou nově načtená data uložena do posledního řádku matice dat.

V dalším kroku je zavolána funkce `zobraz_data`, která slouží k vykreslení měřeného průběhu a také funkce `info`, zajišťující zápis efektivní hodnoty a rozdílu špička-špička měřené veličiny do informačního pole GUI.

Pokud dojde k stisku tlačítka „FFT“ v okamžiku, kdy už je stisknuto tlačítko „Run/Stop“, proběhne následující sekvence příkazů:
nejprve je příkazem `set(handles.vykonej_fft, 'BackgroundColor','b')` nastavena modrá barva tlačítka „FFT“, je nastaven stav tlačítka „Spekt“ na nestlačeno (příkaz `set`, vlastnost `Value` na `0)`, nastavena barva tlačítka „Spekt“ na zelenou a zavolána funkce `ukaz_FFT` (provede vykreslení průběhu FFT v definovaném okně). Po ukončení stisku tlačítka „FFT“ dojde k jeho přebarvení na barvu zelenou.

Pokud dojde k stisku tlačítka Spekt v době, kdy je zároveň stisknuto tlačítko „Run/Stop“, proběhne následující sekvence příkazů:
nejprve je příkazem `set(handles. spektrogram, 'BackgroundColor','b')` nastavena modrá barva tlačítka „Spekt“, je nastaven stav tlačítka „FFT“ na nestlačeno `(set(handles. vykonej_fft, 'Value',0))`, nastavena barva tlačítka „FFT“ na zelenou a zavolána funkce `zobraz_spektrum` (provede vykreslení spektrogramu v okně). Po ukončení stisku tlačítka „Spekt“ dojde k jeho přebarvení na zelenou barvu.

Při čtení dat je v každém kroku cyklu `while - end` testován stav (stisk) tlačítka „Run/Stop“. Při zjištění, že tlačítko „Run/Stop“ není aktivní (`handles.c = 0`), je nastavena stav tlačítek („FFT“, „Spekt“ a „Run/Stop“) na výchozí stav – nestisknuto a rovněž je přiřazena barva pozadí tlačítek na zelenou. Příkazem `cla(handles.zobraz_FFT,'reset')` je provedeno vymazání vykresleného průběhu (FFT nebo spektrogramu) a závěrem funkce je GUI příkazem `guidata` uloženo.

```
function autoset_Callback(hObject, eventdata, handles)
handles.probehlo_cteni_dat =handles.probehlo_cteni_dat + 1; %budu cist data
invoke(handles.deviceObj, 'autoset');%autoset
set(handles.edit_casova_zakladna, 'String',…
num2str(get(handles.deviceObj.Acquisition (1), 'Timebase')));
caszakl=str2double(get(handles.edit_casova_zakladna, 'String'));
vypocet = (1/caszakl)*handles.popup; %do pozice popup menu, kterou hledam,
dostanu vypoctem 1
index=findstr (vypocet,1); % do promenne index je nactena hledana pozice
(cislo 1)
```

```
set(handles.vyber_casova_zakladna, 'Value',index); %aktualizace popup menu
if handles.kan == 'channel1'
set(handles.volt_dilek_ch1,'String',…
num2str(get(handles.deviceObj.Channel(1),'Scale')));
retez1 = get(handles.volt_dilek_hrube_ch1, 'String');
hodn1 = str2double(retez1{get(handles.volt_dilek_hrube_ch1, 'Value')});
val1=str2double(get(handles.volt_dilek_ch1,'String'))/hodn1;
set(handles.volt_dilek_jemne_ch1,'Value',val1);
else
set(handles.volt_dilek_ch2,'String',…
num2str(get(handles.deviceObj.Channel(2),'Scale')));
retez2 = get(handles.volt_dilek_hrube_ch2, 'String');
hodn2 = str2double(retez2{get(handles.volt_dilek_hrube_ch2, 'Value')});
val2=str2double(get(handles.volt_dilek_ch2,'String'))/hodn2;
set(handles.volt_dilek_jemne_ch2,'Value',val2);
end
[handles.X, handles.Y, fft_data] = jednorazove_nacteni(hObject, eventdata,
handles);
handles.fft_data=fft_data;
zobraz_data(hObject, eventdata, handles);
info(hObject, eventdata, handles);
guidata(hObject, handles);
```

Funkce `autoset_Callback` slouží k optimalizaci nastavení Tektronixu vzhledem k měřené veličině. Nejprve je inkrementována hodnota pomocné proměnné `handle.probehlo_nacteni_dat`, protože v průběhu této funkce dochází vždy ke komunikaci počítače s osciloskopem a je tedy zaručeno načtení vstupních dat. Příkazem `invoke(handles.deviceObj,'autoset')` dojde k automatickému nastavení parametrů přístroje a optimalizaci zobrazení měřeného signálu. Z důvodu změny nastavení časové základny osciloskopu je v dalším kroku příkazem `set` nastavena do editačního pole časové základny aktuální hodnota základny (ta je zjištěna příkazem `get(handles.deviceObj.Acquisition(1),'Timebase')` a příkazem `str2double` převedena z řetězce na číslo). Do proměnné `caszakl` je dále načtena aktuální hodnota časové základny, v následném kroku je do proměnné `vypocet` uložen podíl proměnné `handles.popup` a hodnoty časové základny. Smyslem této matematické operace je výpočet matice právě s jedinou hodnotou 1, pozice hodnoty odpovídá pozici nastavené časové základny v proměnné (matici) `handles.popup`. Po vyhledání 1 (hledáme příkazem `findstr`), je hodnota indexu uložena do položky `Value` ve výběrovém (pop-up) menu časové základny.

Tímto způsobem je zajištěno provázání editačního pole časové základny s výběrovým menu. V dalším kroku je zjištěn zdroj signálu (čtení z kanálu č. 1 nebo kanálu č. 2) a podle hodnoty vstupu je provedena aktualizace příslušného editačního pole nastavení V/dílek a příslušného posuvníku (pro kanál č. 1 nebo kanál č. 2) s nastavením Tektronixu. Následným příkazem `[handles.X, handles.Y, fft_data] = jednorazove_nacteni(hObject, eventdata, handles)` jsou do proměnných `handles.X` a `handles.Y` načtena data a do proměnné `fft_data` data odpovídající Fourierově transformaci měřeného průběhu. Následně je zavolána funkce `zobraz_data`, která slouží k vykreslení měřeného průběhu, dále funkce `info`, která zabezpečí v informačním poli výpis efektivní hodnoty a rozdílu špička-špička signálu a v závěru funkce je uloženo GUI.

```
function konec_Callback(hObject, eventdata, handles)
close all
```

Po zavolání funkce `konec` dojde k uzavření všech aktivních oken.

```
function vykonej_fft_Callback(hObject, eventdata, handles)
if handles.probehlo_cteni_dat == 0
pomoc(hObject, eventdata, handles);
pause(10);
end
if handles.c == 0 %FFT pokud neni stlaceno tlacitko Run/Stop
four= get(handles.vykonej_fft, 'Value'); %stav tlacítka FFT
if four ==1
set(handles.vykonej_fft, 'BackgroundColor','b'); %barva tlacitka FFT (modre) -
je stlaceno
set(handles.spektrogram, 'Value',0); %nastav tlacítka spekt nezmácknuto
set(handles.spektrogram, 'BackgroundColor','g'); %barva tlacitka Spekt
(zelene) - neni stlaceno
ukaz_FFT(hObject, eventdata, handles);
else
set(handles.vykonej_fft, 'BackgroundColor','g'); %barva tlacitka FFT (zelene)
- neni stlaceno
cla(handles.zobraz_FFT,'reset');
end
end
guidata(hObject, handles);
```

Funkce `vykonej_fft_Callback` slouží k zobrazení Fourierovy transformace měřené veličiny. Nejprve proběhne kontrola, zda jsou již načtena vstupní data, pokud tomu tak není, je zobrazeno v popředí obrazovky okno s nápovědou a přerušeno vykonání programu na dobu 10 vteřin (uživatel tak dostává časový prostor pro načtení dat).

Pokud data jsou načtena, je provedena kontrola, zda je stlačeno tlačítko „Run/Stop“ (tento stav je ošetřen ve funkci `ukaz_FFT`), je provedena kontrola stavu tlačítka „FFT“. Pokud je stav tlačítka „FFT“ slačeno (hodnota pomocné proměnné `four` je 1) je nastaven stav tlačítka „Spekt“ na nestisknuto, barva pozadí tlačítka „FFT“ se změněna na modrou, barva tlačítka „Spekt“ na zelenou.

Následně je zavolána funkce `ukaz_FFT`, při které je zobrazen FFT průběh měřené veličiny. Po ukončení stisku tlačítka „FFT“ je barva pozadí tlačítka změněna na zelenou, stonkový graf FFT průběhu je smazán a celé GUI je uloženo.

```
function spektrogram_Callback(hObject, eventdata, handles)
if handles.probehlo_cteni_dat == 0
pomoc(hObject, eventdata, handles);
pause(10);
end
if handles.c == 0 %tlacitko Run/Stop neni zmacknuto
tl_sp= get(handles.spektrogram, 'Value'); %stav tlacítka spek
set(handles.vykonej_fft, 'Value',0); %nastav tlacítka FFT nezmácknut
set(handles.vykonej_fft, 'BackgroundColor','g'); %barva tlacitka FFT (zelene)
- neni stlaceno
if tl_sp==1
set(handles.spektrogram, 'BackgroundColor','b'); %barva tlacitka Spek (modra)
- je stlaceno
zobraz_spektrum(hObject, eventdata, handles);
else
set(handles.spektrogram, 'BackgroundColor','g'); %barva tlacitka Spekt
(zelene) - neni stlaceno
cla(handles.zobraz_FFT,'reset'); %smaze aktuální handles
end
end
guidata(hObject, handles);
```

Funkce `spektrogram_Callback` slouží k zobrazení spektrogramu měřené veličiny. Nejprve proběhne kontrola, zda jsou již načtena vstupní data, pokud tomu tak není, je zobrazeno

okno s nápovědou a přerušeno vykonání programu na dobu 10 vteřin (uživatel tak opět dostává časový prostor pro načtení dat).

Pokud data jsou načtena, je provedena kontrola, zda tlačítko „Run/Stop“ je stlačeno (tento stav je ošetřen ve funkci `zobraz_spektrum`), je provedena kontrola stavu tlačítka „Spekt“. Pokud je stav tlačítka „Spekt“ stlačeno (hodnota pomocné proměnné `tl_sp` je 1) je nastaven stav tlačítka „FFT“ na nestisknuto, barva pozadí tlačítka „Spekt“ je změněna na modrou, tlačítka „Spekt“ na zelenou.

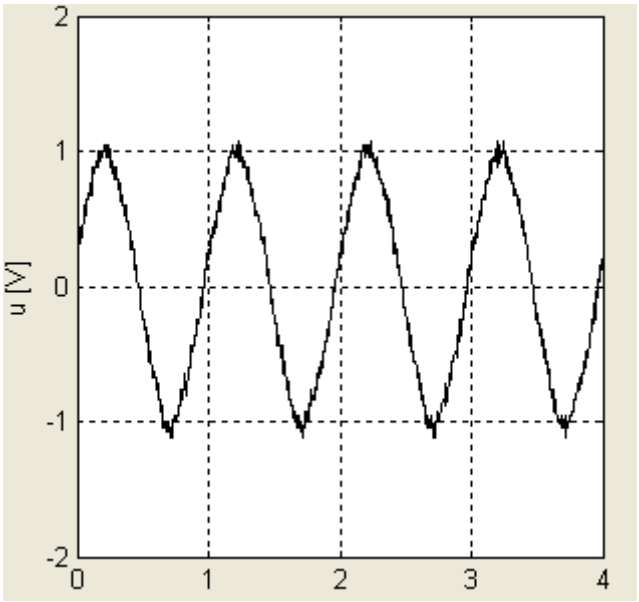
Následně je zavolána funkce `zobraz_spektrum` při níž je zobrazen spektrogram průběhu měřené veličiny. Po ukončení stisku tlačítka „Spekt“ je barva pozadí tlačítka změněna na zelenou, 3-D stonkový graf je smazán a celé GUI je uloženo.

Funkce pro zobrazení dat

```
function zobraz_data(hObject, eventdata, handles) %zobrazení dat (vykreslení
prubehu po zmene nebo pri stisku Run/Stop)
if handles.probehlo_cteni_dat == 0
pomoc(hObject, eventdata, handles);
end
axes(handles.vykresli_prubeh);
if handles.kan == 'channel1'
krok_Y = get(handles.deviceObj.Channel(1), 'Scale'); %krok je hodnota V/dilek
else
krok_Y = get(handles.deviceObj.Channel(2), 'Scale');
end
max_Y=2*krok_Y; %maximum y-ove osy odpovida dvojnasobku nastaveni rozsahu
Tektronixu
min_Y=-2*krok_Y; %minimum y-ove osy odpovida dvojnasobku nastaveni rozsahu
Tektronixu
xmin=0; %minimum x
xmax=4*(get(handles.deviceObj.Acquisition(1), 'Timebase')); %maximum x
odpovídá ctyrnasobku nastaveni casove zakladny Tektronixu
plot(handles.X(end,:), handles.Y(end,:), 'k'), grid on; %vykresleni prubehu,
mrizka aktivni
axis([xmin xmax min_Y max_Y]); %nastaveni hranicnich bodu os dle pozadavku
set(gca,'YTick',min_Y:krok_Y:max_Y); %nastaveni y-ove osy
xlabel ('t [s]'); %nazev x-ove osy
ylabel ('u [V]'); %nazev y=ove osy
```

Funkcí `zobraz_data` je definováno zobrazení dat, která jsou načtena při komunikaci PC s osciloskopem Tektronix. Nejprve je v běhu programu otestováno, zda již k načtení dat z osciloskopu došlo, v negativním případě je zobrazeno okno upozorňující na nenačená data. Příkazem `axes(handles.vykresli_prubeh)` je definováno, že data budou vykreslena do požadovaného grafického objektu (okno `vykresli_prubeh`).

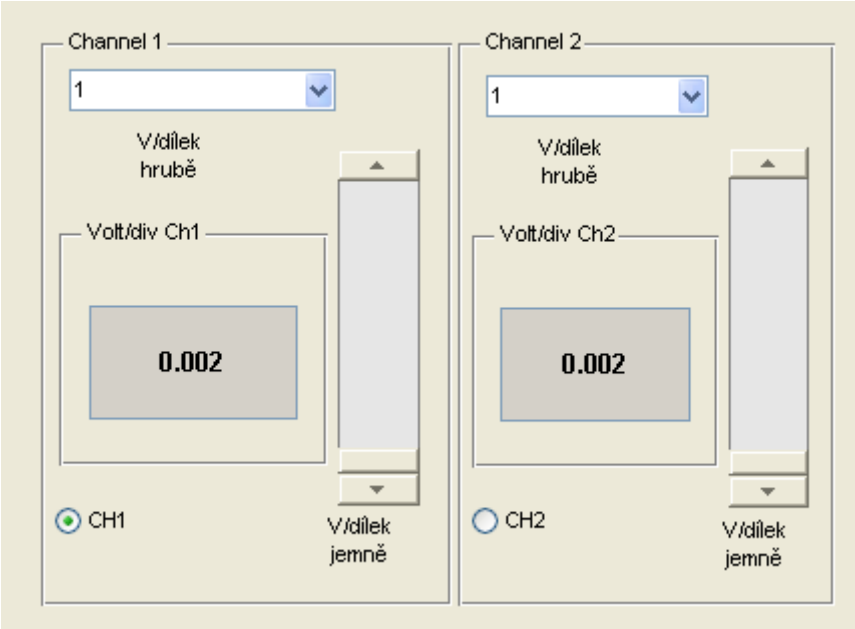
Dílčí krok pro popis osy y je definován jako hodnota rozlišení V/dílek.V dalším kroku je pro regulérní zobrazení popisu os definováno minimum osy y (`min_Y=-2*krok_Y`) a maximum (`max_Y=2*krok_Y`), minimum osy x je 0, maximum je čtyřnásobek hodnoty nastavené časové základny osciloskopu. Příkazem `plot` dojde k vykreslení posledního řádku naměřených dat `handles.X(end,:)` pro x-ovou složku a `handles.Y(end,:)` pro y-ovou složku dat. Parametr '`k`' v příkazu `plot` určuje barvu pro vykreslení - černou (blac**k**), příkaz `grid on` nastaví aktivní zobrazení mřížky v grafu. Definování mezních bodů jednotlivých os je provedeno příkazem `axis([xmin xmax min_Y max_Y])`. Příkaz `xlabel('t[s]')`, respektive `ylabel('u[V]')` definuje text, kterým budou jednotlivé osy popsány. Průběh naměřené veličiny je patrný z obrázku č. 19.



Obrázek č. 19: Okno se zobrazení naměřeného průběhu

Funkce pro výběr kanálu jeho zobrazení

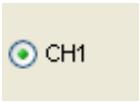
Výběr zdroje pro čtení dat (kanál č. 1 nebo 2) a hodnotu zobrazení V/dílek je možno v GUI ovládat pomocí dvou radiobuttonů, dvou výběrových menu, dvou postníků a dvou editačních políček, které se navzájem doplňují (viz obrázek č. 20).



Obrázek č. 20: Ovládací panel pro ovládání kanálu a jeho zobrazení

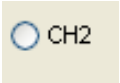
Tlačítko předvolby

```
function nastav_ch1_Callback(hObject, eventdata, handles)
set(handles.nastav_ch1,'Value', 1); %tlacitko CH1 zvoleno
set(handles.nastav_ch2, 'Value',0); %tlacitko CH2 vypnuto
handles.kan = 'channel1';
set(handles.deviceObj.Measurement(1), 'Source','channel1');
guidata(hObject, handles);
```



Funkcí `nastav_ch1_Callback` měníme vlastnosti grafických objektů radiobuttonů (tlačítek předvolby) na stav vybrán nebo nevybrán - příkazem `set`, nastavením vlastnosti ‘Value‘ na hodnotu 0 (nevybráno – pro kanál č. 2) a 1 (vybrán kanál č. 1). Do proměnné `handles.kan` je načten řetězec `channel2`, následně v dalším kroku je příkazem `set` nastaven zdroj čtení `Source` na kanál č. 1. Příkaz `guidata` uloží grafický objekt.

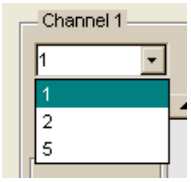
```
function nastav_ch2_Callback(hObject, eventdata, handles)
set(handles.nastav_ch1,'Value', 0); %tlacitko CH1 vypnuto
set(handles.nastav_ch2, 'Value',1); %tlacitko CH2 zvoleno
handles.kan = 'channel2';
set(handles.deviceObj.Measurement(1), 'Source', 'channel2');
guidata(hObject, handles);
```



Popis funkce `nastav_ch2_Callback` je analogický s popisem předchozí funkce, obě funkce jsou vůči sobě inverzní (vždy je vybrán pouze jeden kanál pro čtení dat).

Výběrové menu

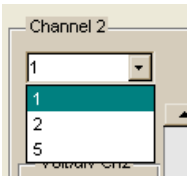
```
function volt_dilek_hrube_ch1_Callback(hObject, eventdata, handles
retez1 = get(handles.volt_dilek_hrube_ch1, 'String');
hodn1 = str2double(retez1{get(handles.volt_dilek_hrube_ch1, 'Value')});
if hodn1 > 1
set(handles.deviceObj.Channel(1),'Scale', hodn1);
set(handles.volt_dilek_ch1,'String',num2str(get(handles.deviceObj.Channel(1),'
Scale')));
else
set(handles.deviceObj.Channel(1), 'Scale', 0.1);
set(handles.volt_dilek_ch1,'String',num2str(get(handles.deviceObj.Channel(1),'
Scale')));
end
if handles.c == 0 & max(max(handles.X))~=0
zobraz_data(hObject, eventdata, handles);
end
guidata(hObject, handles);
```



Funkce `volt_dilek_hrube_ch1_Callback` slouží k výběru hodnoty z výběrového menu (nastavení zobrazení V/dílek – skoková změna). Nejprve je do proměnné `retez1` načtena hodnota výběrového menu V/dílek ve formátu řetězce, ten je následně převeden na číslo, které je uloženo do proměnné `hodn1`. Pokud je proměnná `hodn1` větší než 1 (ve výběrovém menu jsou předdefinovány hodnoty 1, 2, 5 V/dílek), je do vlastnosti `Scale` (hodnota rastru zobrazení) osciloskopu vložena hodnota proměnné `hodn1` (vybraná volba z rozbalovacího menu). V případě, že je vybrána hodnota 1, je do vlastnosti `Scale` osciloskopu vložena hodnota 0,1 V/dílek.

Následně proběhne test stisku tlačítka „Run/Stop“, otestuje se, zda jsou naměřena data, případné vykreslení grafu průběhu naměřeného signálu a uložení grafického rozhraní.

```
function volt_dilek_hrube_ch2_Callback(hObject, eventdata, handles)
retez2 = get(handles.volt_dilek_hrube_ch2, 'String');
hodn2 = str2double(retez2{get(handles.volt_dilek_hrube_ch2, 'Value')});
if hodn2 > 1
set(handles.deviceObj.Channel(2),'Scale', hodn2);
set(handles.volt_dilek_ch2,'String',num2str(get(handles.deviceObj.Channel(2),'
Scale')));
else
set(handles.deviceObj.Channel(2), 'Scale', 0.1);
set(handles.volt_dilek_ch2,'String',num2str(get(handles.deviceObj.Channel(2),'
Scale')));
end
if handles.c == 0 & max(max(handles.X))~=0
zobraz_data(hObject, eventdata, handles);
end
guidata(hObject, handles);
```



Popis funkce `volt_dilek_hrube_ch2` lze odvodit z funkce `volt_dilek_hrube_ch1`, tato je popsána výše.

Posuvník

```
fiction volt_dilek_jemne_ch1_Callback (hObject, eventdata, handles)
retez1 = get(handles.volt_dilek_hrube_ch1, 'String');
hodn1 = str2double(retez1{get(handles.volt_dilek_hrube_ch1, 'Value')});
set(handles.volt_dilek_ch1,'String',...
num2str(get(handles.volt_dilek_jemne_ch1,'Value')*hodn1));
set(handles.deviceObj.Channel(1),'Scale',
str2double(get(handles.volt_dilek_ch1,'String')));
set(handles.volt_dilek_ch1,'String',num2str…
(get(handles.deviceObj.Channel(1),'Scale')));
if handles.c == 0 & max(max(handles.X))~=0
zobraz_data(hObject, eventdata, handles);
end
```

guidata(hObject, handles);

Funkce `volt_dilek_jemne_ch1_Callback` slouží k ovládání posuvníku pro kanál č. 1 (jemné ovládání zobrazení Volt/dílek v rozsahu minimum 0,002 až hodnota nastavená výběrovým menu). Nejprve je do proměnné `retez1` načtena hodnota výběrového menu V/dílek (hrubé nastavení), následně je převeden řetězec na číselnou hodnotu a uložena do proměnné `hodn1`. Následným příkazem `set` je do editačního pole nastavení V/dílek nastavena

hodnota, která přesně odpovídající poloze posuvníku, zjištěná příkazem `get(handles.volt_dilek_jemne_ch2,'Value')`, vynásobená hodnotou hrubého nastavení V/dílek, tedy hodnotou vybranou z výběrového menu. Algoritmus násobení je využit proto, neboť polohu posuvníku je implicitně možno popsat číslem v intervalu <0, 1>, kde minimum odpovídá 0 a 1 odpovídá maximu. Hodnotu maxima i minima lze sice kdykoli upravit ve vlastnostech grafického objektu posuvníku například v Layout editoru, v průběhu programu upravujeme maximum posuvníku průběžně podle potřeby násobením číslem odpovídajícím hrubému nastavení V/dílek (násobení hodnotou 1, 2 nebo 5).

Následně je hodnota požadovaného rozlišení V/dílek nastavena do osciloskopu. V rámci zpětné vazby (korekce případné odchylky požadavku na nastavení osciloskopu a skutečně nastavené hodnoty) je z osciloskopu příkazem `get` získána hodnota `Scale`, převedena z číselné podoby na řetězec a uložena do editačního pole nastavení V/dílek kanálu č. 1.

V posledním fázi programu je provedeno testování, zda je či není stisknuto tlačítko „Run/Stop“ a zároveň, zda matice dat není prázdná (maximální hodnota dat matice `handles.X` je různá od nuly). Pokud je podmínka splněna, je zavolána funkce zobrazení dat `zobraz_data`. Příkaz `guidata` zajistí uložení GUI.

```
function volt_dilek_jemne_ch2_Callback(hObject, eventdata, handles)
retez2 = get(handles.volt_dilek_hrube_ch2, 'String');
hodn2 = str2double(retez2{get(handles.volt_dilek_hrube_ch2, 'Value')});
set(handles.volt_dilek_ch2,'String',...
num2str(get(handles.volt_dilek_jemne_ch2,'Value')*hodn2));
set(handles.deviceObj.Channel(2),'Scale',…
str2double(get(handles.volt_dilek_ch2,'String')));
set(handles.volt_dilek_ch2,'String',num2str…
(get(handles.deviceObj.Channel(2),'Scale')));
if handles.c == 0 & max(max(handles.X))~=0
zobraz_data(hObject, eventdata, handles);
end
guidata(hObject, handles);
```

Funkce `volt_dilek_jemne_ch2_Callback` slouží k k ovládání posuvníku pro kanál č. 2 a popis funkce je totožný s popisem funkce `volt_dilek_jemne_ch1`, pouze názvy proměnných jsou označeny symboly s číslem 2 v názvu a veškeré operace se váží ke kanálu č.2.

Editační pole

```
function volt_dilek_ch1_Callback(hObject, eventdata, handles)
retez1 = get(handles.volt_dilek_hrube_ch1, 'String');
hodn1 = str2double(retez1{get(handles.volt_dilek_hrube_ch1, 'Value')});
val1=str2double(get(handles.volt_dilek_ch1,'String'))/hodn1;
if isnumeric(val1) & length(val1)==1 &...
val1 >= get (handles.volt_dilek_jemne_ch1,'Min') &...
val1 <= get (handles.volt_dilek_jemne_ch1,'Max')
set(handles.volt_dilek_jemne_ch1,'Value',val1);
set(handles.deviceObj.Channel(1),'Scale',…
str2double(get(handles.volt_dilek_ch1,'String')));
set(handles.volt_dilek_ch1,'String',num2str…
(get(handles.deviceObj.Channel(1),'Scale')));
val1=str2double(get(handles.volt_dilek_ch1,'String'));
set(handles.volt_dilek_ch1,'Value',val1);
else
errordlg('Zadej znovu cislo v rozsahu','Chyba zadani');
pause(3);
end
if handles.c == 0 & max(max(handles.X))~=0
zobraz_data(hObject, eventdata, handles);
end
guidata(hObject, handles);
```

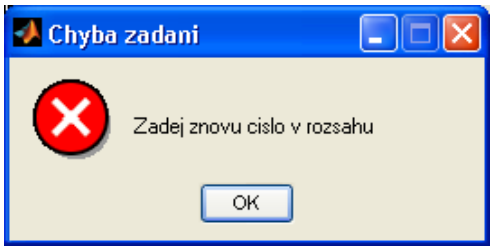
Funkce `volt_dilek_ch1_Callback` umožňuje zadat požadovanou hodnotu zobrazení V/dílek pro kanál č. 1 zápisem do editačního pole. Do proměnné `retez1` je nejprve načtena hodnota hrubého nastavení V/dílek, tedy hodnota výběrového menu, ta je příkazem `str2double` převedena na číslo a uložena do pomocné proměnné `hodn1`. Do proměnné `val1` je následně vypočtena hodnota, odpovídající podílu čísla zapisovaného do editačního pole a hodnoty hrubého nastavení V/dílek (vybraná hodnota výběrového menu). Po kontrole, že v proměnné `val1` je uloženo číslo a zároveň proměnná `val1` je v rozmezí mezi minimální a maximální hodnotou posuvníku:

- je nastavena poloha posuvníku (vlastnost `Value`, odpovídá hodnotě mezi 0,002 a 1 a odpovídá podílu, uloženým v proměnné `val1`),

- je nastaveno rozlišení V/dílek osciloskopu (vlastnost `Scale`),

- je aktualizováno editační pole (`handles.volt_dilek_ch1` a její vlastnost `String`) podle nastavení osciloskopu (`get(handles.deviceObj.Channel(1),'Scale')`.

V případě, že je zadání v editačním poli nesprávné (např. záporné číslo, číslo větší než jedna při výběru hrubého zobrazení 1 v popup menu, číslo větší než 2 v případě volby 2 v popup menu apod), je zobrazeno dialogové okno v grafickém provedení chyba MATLABu s textem `Zadej znovu cislo v rozsahu`. Běh programu je v tomto případě přerušen na 3 vteřiny tak, aby bylo zajištěno, že se okno s výstrahou nedostane ihned do pozadí a bylo uživatelem zaregistrováno. Grafická podoba varování je zobrazena na níže uvedeném obrázku č. 21:



Obrázek č. 21: Chybové okno – pokyn pro změnu vstupní hodnoty

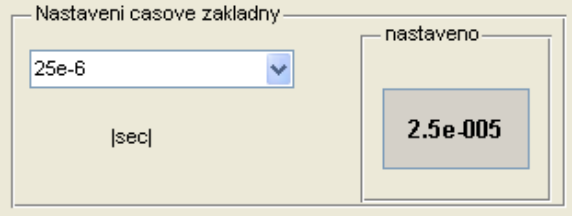
Dále následuje testování stisku tlačítka „Run/Stop“, kontrola načtení dat a případné vykreslení dat a uložení grafického rozhraní.

```
function volt_dilek_ch2_Callback(hObject, eventdata, handles)
retez2 = get(handles.volt_dilek_hrube_ch2, 'String');
hodn2 = str2double(retez2{get(handles.volt_dilek_hrube_ch2, 'Value')});
val2=str2double(get(handles.volt_dilek_ch2,'String'))/hodn2;
if isnumeric(val2) & length(val2)==1 &...
val2 >= get (handles.volt_dilek_jemne_ch2,'Min') &...
val2 <= get (handles.volt_dilek_jemne_ch2,'Max')
set(handles.volt_dilek_jemne_ch2,'Value',val2);
set(handles.deviceObj.Channel(2),'Scale',…
str2double(get(handles.volt_dilek_ch2,'String')));
set(handles.volt_dilek_ch2,'String',num2str(get(handles.deviceObj.Channel(2),'
Scale')));
val2=str2double(get(handles.volt_dilek_ch2,'String'));
set(handles.volt_dilek_ch2,'Value',val2);
else
errordlg('Zadej znovu cislo v rozsahu','Chyba zadani');
pause(3);
end
```

```
if handles.c == 0 & max(max(handles.X))~=0
zobraz_data(hObject, eventdata, handles);
end
guidata(hObject, handles);
```

Funkce `volt_dilek_ch2_Callback` je identická s funkcí pro editaci kanálu č. 1, pouze příslušné proměnné a grafické prvky mají vazbu na kanál č. 2.

Funkce pro nastavení časové základny



Obrázek č. 22: Editační okno pro nastavení časové základny

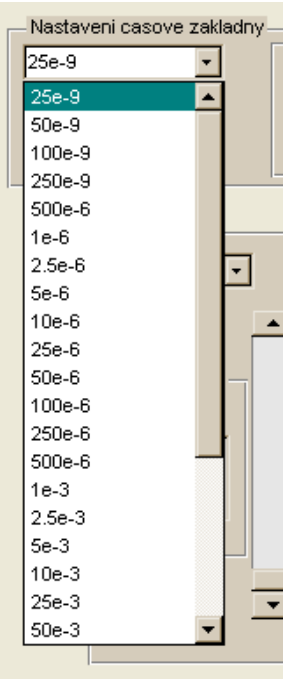
Výběrové menu

```
function vyber_casova_zakladna_Callback(hObject, eventdata, handles)
retezec = get(handles.vyber_casova_zakladna, 'String');
hodnota = str2double(retezec{get(handles.vyber_casova_zakladna, 'Value')});
set(handles.deviceObj.Acquisition(1), 'Timebase', hodnota);
set(handles.edit_casova_zakladna, 'String',
num2str(get(handles.deviceObj.Acquisition (1), 'Timebase')));
if handles.c == 0 & max(max(handles.X))~=0
zobraz_data(hObject, eventdata, handles);
end
guidata(hObject, handles);
```

Funkce `vyber_casova_zakladna_Callback` zajišťuje výběr hodnoty časové základny z výběrového menu. Do proměnné `retezec` je příkazem `get` z výběrového menu vybrán řetězec odpovídající požadované (vybírané) hodnotě z přednastavené nabídky. V dalším kroku je řetězec převeden příkazem `str2double` do číselné podoby a uložen do proměnné `hodnota`. Následným příkazem `set` je do vlastnosti časová základna `Timebase` uložena hodnota proměnné `hodnota`. Pro zpětnou aktualizaci editačního pole časové základny je

v dalším příkazu `set(handles.edit_casova_zakladna,'String',num2str…(get(handles.deviceObj.Acquisition(1),'Timebase')))` zjištěna hodnota vlastnosti `Timebase` osciloskopu, převedena z číselné podoby na řetězec a vložena do editačního pole časové základny.

Následně je proveden test, zda není stisknuto tlačítko „Run/Stop“ a zároveň zda není prázdná matice dat (maximum matice dat `handles.X` je různé od nuly). Pokud je podmínka splněna, je zavolána funkce zobrazení dat `zobraz_data`. Příkaz `guidata` uloží grafické uživatelské rozhraní. Výběrové menu je zobrazeno na obrázku č. 23.



Obázek č. 23: Nastavení časové základny výběrem

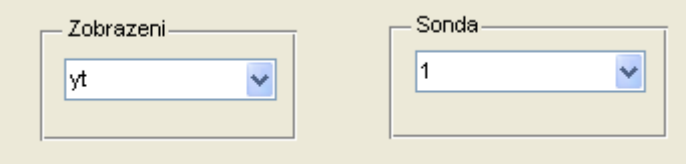
Editační pole

```
function edit_casova_zakladna_Callback(hObject, eventdata, handles)
if isnumeric(handles.edit_casova_zakladna)
caszakl=str2double(get(handles.edit_casova_zakladna, 'String'));
if caszakl <= 25.0E-9
caszakl = 25.0E-9;
end
if caszakl >= 50
caszakl = 50;
end
set(handles.deviceObj.Acquisition(1), 'Timebase', caszakl); %nastaveni cas. zakladny dle ediacniho pole
set(handles.edit_casova_zakladna,'String',
num2str(get(handles.deviceObj.Acquisition(1), 'Timebase')));
caszakl=str2double(get(handles.edit_casova_zakladna, 'String')); %aktualizace promenne caszakl
vypocet = (1/caszakl)*handles.popup; %do polohy popup menu, kterou hledam, dostanu vypoctem 1
index=findstr (vypocet,1); %do promenne index je nactena hledana poloha
set(handles.vyber_casova_zakladna, 'Value',index); %aktualizace popup menu
end
if handles.c == 0 & max(max(handles.X))~=0
zobraz_data(hObject, eventdata, handles);
end
guidata(hObject, handles);
```

Funkce `edit_casova_zakladna_Callback` slouží k nastavení časové základny prostřednictvím editačního pole. Nejprve se příkazem `isnumeric` otestuje, zda je do editačního pole zadávána číselná hodnota. V případě, že je podmínka splněna, je získána z proměnné `handles.edit_casova_zakladna` hodnota (ve formátu řetězec a dále převedena na číslo) a následně uložena do proměnné `caszakl`. Pro eliminaci zadání mimo rozsah hodnot 250 µs až 50 s jsou hodnoty menší než 250 µs nahrazeny hodnotou 250 µs, obdobně hodnoty větší než 50 s jsou přepsány hodnotou 50s (do proměnné `caszakl`). Následně je hodnota proměnné `caszakl` vložena příkazem `set` do osciloskopu (vlastnost `Timebase`). Zpětně je ověřeno, jaká je hodnota časové základny uložena v osciloskopu (příkaz `get(handles.deviceObj.Acquisition(1),'Timebase')`), převedena pomocí `num2str` na text a vložena do editačního pole (tento postup je nutný z toho důvodu, neboť např. při požadavku na nastavení hodnoty časové základny 0,987654321 se do osciloskopu

fyzicky vloží hodnota 1,0 a tuto je třeba zpětně zapsat do editačního pole a nahradit tak neakceptovatelnou položku). V dalším kroku je aktualizována hodnota proměnné `caszakl` příkazem `get` z `handles.edit_casova_zakladna`. Do proměnné `vypocet` (matice nenulových dat s jedinou hodnotou odpovídající hodnotě časové základny) získáme výpočtem (podílem hodnotou odpovídající nastavení časové základny) data s jediným číslem jedna. Právě hodnotu 1 v dalším kroku hledáme a „polohu“ tohoto prvku uložíme do proměnné `index`. Hodnota proměnné `index` následně odpovídá položce ve výběrovém menu, tato je příkazem `set` nastavena (provázání editačního pole časové základny s výběrovým menu časové základny). Následně již probíhá kontrola stisku tlačítka „Run/Stop“ a načtených dat a následné vykreslení nebo nevykreslení a uložení GUI.

Funkce pro nastavení formátu zobrazení a násobku sondy



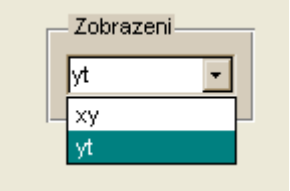
Obrázek č. 24: Výběrovémenu pro formát zobrazení a násobek sondy

Výběrové menu

```
function format_zobrazeni_Callback(hObject, eventdata, handles)
zobr = get(handles.format_zobrazeni, 'String');
vyb_zobr = zobr{get(handles.format_zobrazeni, 'Value')};
set(handles.deviceObj.Display(1), 'Format',vyb_zobr);
if handles.c == 0 & max(max(handles.X))~=0
zobraz_data(hObject, eventdata, handles);
end
guidata(hObject, handles);
```

Funkce `format_zobrazeni_Callback` (na obrázku č. 25) umožňuje zvolit formát zobrazení dat. Jedná se o výběr z výběrového menu, kde příkazem `get` je získán řetězec hodnot zobrazení, ten je uložen do proměnné `zobr`, následně je do proměnné `vybr_zobr` načtena hodnota, vybraná z výběrového menu a do osciloskopu uložen požadavek na změnu zobrazení

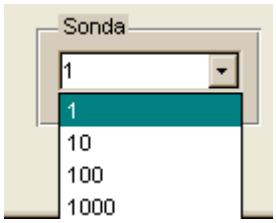
(vlatnost `Format`). Funkce je ukončena testem stisku tlačítka „Run/Stop“, případným vykreslením dat a uložením rozhraní.



Obrázek č. 25: Výběr způsobu zobrazení

```
function vyber_rozsah_sonda_Callback(hObject, eventdata, handles)
kan = get(handles.deviceObj.Measurement(1), 'Source');
if kan == 'channel1'
retez_sonda = get(handles.vyber_rozsah_sonda, 'String');
hodnota_sonda = str2double(retez_sonda{…
get(handles.vyber_rozsah_sonda, 'Value')});
set(handles.deviceObj.Channel(1), 'Probe', hodnota_sonda);
else
retez_sonda = get(handles.vyber_rozsah_sonda, 'String');
hodnota_sonda = str2double(retez_sonda…
{get(handles.vyber_rozsah_sonda, 'Value')});
set(handles.deviceObj.Channel(2), 'Probe', hodnota_sonda);
end
if handles.c == 0 & max(max(handles.X))~=0
zobraz_data(hObject, eventdata, handles);
end
guidata(hObject, handles);
```

Funkce `vyber_rozsah_sonda_Callback` (obrázek č. 26) slouží k nastavení násobku rozsahu měřící sondy. Jedná se o klasické rozbalovací okno, ze kterého je možno vybrat jednu z hodnot 1x, 10x, 100x, 1000x. Princip činnosti této funkce je obdobný jako např. u funkce pro výběr formátu zobrazení, jen na počátku funkce je proveden testt, zda je sonda zapojena na kanál č. 1 nebo č. 2 (zjištění vlastnosti `Source` objektu osciloskopu).



Obrázek č. 26: Výběr rozsahu sondy

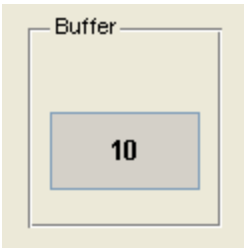
Funkce pro editaci velikosti matice dat

Editační pole

```
function edit_buffer_Callback(hObject, eventdata, handles)

set(handles.edit_buffer, 'String');

buf = str2double(get(handles.edit_buffer, 'String'));

handles.buffer = buf;

guidata(hObject, handles);
```

Funkce `edit_buffer_Callback` (obrázek č. 27) slouží k možnosti změnit počet řádků matice dat zápisem do editačního pole.

Do pomocné proměnné `buf` je načten řetězec z editačního pole (vlastnost `String` proměnné `handles.edit buffer`), převedený na číslo příkazem `str2double`. Následně je proměnná `buf` uložena do proměnné `handles.buffer`. Při startu programu je do proměnné `edit_buffer` načtena hodnota 10, což je s ohledem na zkušenost při testování programu postačující. Při vyšších hodnotách této proměnné dojde sice k uložení většího množství naměřených dat, ale při nedostatečném hardwarovém vybavení počítače (procesor, operační paměť) může dojít při zpracování dat k zahlcení počítače.



Obázek č. 27: Editační pole nastavení bufferu

Funkce reagující na stisk myši v poli zobrazení

```
function zobraz_FFT_ButtonDownFcn(hObject, eventdata, handles)
four= get(handles.vykonej_fft, 'Value'); %stav tlacítka FFT
tl_sp= get(handles.spektrogram, 'Value'); %stav tlacítka spek
if four == 1
figure('Name','FFT','NumberTitle','off')
N = length(handles.fft_data(end,:));%pocet vzorku
Spektrum_dvoustranne = (handles.fft_data (end,:))/N;
delta_f = handles.f_N / N;
f_jednostranne = (0 : delta_f : handles.f_N-delta_f); %osa Xová
if mod(N, 2) == 1 %Lichy pocet vzorku
Amplitudove_Spektrum_jednostranne = [abs(Spektrum_dvoustranne(1)), 2*abs
(Spektrum_dvoustranne(2 : N-1 ))];
else
Amplitudove_Spektrum_jednostranne = [abs(Spektrum_dvoustranne(1)), 2*abs
(Spektrum_dvoustranne(2 : N))];
end
fft = stem(f_jednostranne, Amplitudove_Spektrum_jednostranne);
set(fft,'MarkerFaceColor','red'); %nastaveni bodu grafu (cervena)
xlabel('Freq (Hz)') %nazev osy x
ylabel('|Y(f)|') %nazev osy y
end
if tl_sp == 1
figure('Name','Spektrogram','NumberTitle','off')
Buf=size(handles.fft_data, 1); %pocet nactenych radku
N = length(handles.fft_data); %pocet vzorku
delta_f = handles.f_N / N; %krok na ose frekvence
[osa_X, Y_f] = meshgrid (0 : delta_f : handles.f_N-delta_f, 1:size(handles.
fft_data, 1));
for k = 1 : Buf
if mod(N, 2) == 1 %Lichy pocet vzorku - jednostranne spektrum
vektor_fft(k,:) = [abs(handles.fft_data(1))/N, 2*abs(handles.fft_data(2 : N
-1))/N];
else
vektor_fft(k,:) = [abs(handles.fft_data(1))/N, 2*abs(handles.fft_data(2 :
N))/N];
end
end
spek=stem3(osa_X, Y_f, vektor_fft); %graf spektra
```

```
set(spek,'MarkerFaceColor','red'); %nastaveni bodu spektra (cervena)
xlabel('Freq (Hz)') %nazev x-ove osy
ylabel('buffer') %nazev y-ove osy
zlabel('|Y(f)|') %nazev z-osy
end
```

Funkce `zobraz_FFT_ButtonDownFcn` zajišťuje, aby v případě stisku libovolného tlačítka myši na grafu dat FFT nebo spektrogramu došlo k vykreslení identického průběhu v novém okně (obrázek č. 10 a 11). Nejprve je zjištěn stav tlačítek „FFT“ a „Spekt“ (proměnné `four` a `tl_sp`).

Pokud je stisknuto tlačítko „FFT“ (`four = 1`) je otevřeno nové okno s názvem `FFT`, přičemž parametr `'NumberTitle','off'` znamená, že nově otevřené okno nebude číslováno. Do proměnné `N` je uložen počet vzorků měřené veličiny (příkazem `length` je zjištěn počet prvků matice `handles.fft_data`, respektive jejího posledního řádku dat `(end,:)`), do proměnné `delta_f` je vypočítán a uložen krok pro zobrazení osy x grafu a do proměnné `f_jednostranne` jsou uložena data popisující xovou osu (frekvence). Do proměnné `Spektrum_dvoustranne` je uložen výsledek FFT měřené veličiny (FFT posledního řádku dat matice `handles.fft_data`. V následném kroku je zjišťováno, zda je počet vzorků sudé nebo liché číslo (zda je zbytek po dělení modulo 2 roven 0 nebo 1) a podle počtu vzorků je z oboustranného spektra vytvořeno jednostranné spektrum. Jednostranné spektrum je definováno jako absolutní hodnota dvojnásobku hodnoty dvojstranného spektra. Pro lichý počet prvků je poslední prvek matice dat FFT definován jako předposlední vzorek. Následně je do proměnné `fft` uložen stonkový graf s prvky odpovídajícími jednostrannému průběhu FFT.

Příkazem `set(fft,'MarkerFaceColor','red')` je definována červená barva koncových bodů stonkového grafu. Přikazy `xlabel` a `ylabel` jsou definovány popisky os.

Pokud je stisknuto tlačítko „Spekt“ (`tl_sp = 1`) je otevřeno nové okno s názvem `Spektrogram`, přičemž parametr `'NumberTitle','off'` znamená, že nově otevřené okno nebude číslováno. Do proměnné `N` je pomocí příkazu `length` zjištěn počet prvků (délka matice `handles.fft_data`, jejího posledního řádku dat `(end,:)`, což odpovídá počtu vzorků měřené veličiny, do proměnné `delta_f` je vypočítán a uložen krok pro zobrazování osy x grafu. Pro vykreslení průběhu spektrogramu je nutno zobrazit 3-D graf, z tohoto důvodu je nutno definovat z- osu (FFT) v závislosti na osách x a y (frekvence a počet zaplněných řádků matice dat FFT - odpovídá proměnné `Buf`). V následném kroku je zjišťováno, zda je počet vzorků sudé nebo liché číslo (zda je zbytek po dělení modulo 2 roven 0 nebo 1) a podle počtu

vzorků je z oboustranného spektra vytvořeno jednostranné spektrum (jeho absolutní hodnota, pro lichý počet prvků odpovídá poslední prvek předposlednímu vzorku). Následně je do proměnné `spekt` uložen trojrozměrný stonkový graf s prvky odpovídajícími několika po sobě jdoucích jednostranných průběhů FFT. Příkazem `set(spekt,'MarkerFaceColor',…`

`'red')` je definována červená barva koncových bodů 3-D stonkového grafu. Přikazy `xlabel, ylabel` a `zlabel` jsou definovány popisky os.

3. Zjednodušený uživatelský manuál

Aplikace `tektronix,` vytvořená pomocí grafického uživatelského rozhraní GUIDE programu MATLAB, slouží k měření a vykreslení průběhu ustálených nebo pomalu se měnících veličin. Pomocí GUI je rovněž možno ovládat nebo nastavovat vybrané vlastnosti osciloskopu Tektronix TDS 2002B.

Níže je popsán postup pro vytvoření spojení osciloskopu s počítačem a práci s GUI:

Do adresáře, ve kterém se nacházejí soubory `textronix.m` a `tektronix.fig` (nastaveného pracovního adresáře programu MATLAB) vložíme soubor ovladače příslušného osciloskopu (`tektronix_tds2002B.mdd` nebo alternativně `…_tds2000.mdd`).

Po spuštění programu MATLAB navážeme komunikaci mezi PC a osciloskopem[3]. Navázání komunikace PC s osciloskopem lze shrnout do 6 postupných kroků:

1. po spuštění MATLABu zadáme příkaz `tmtool,`
2. připojíme osciloskopu k počítači pomocí rozhraní USB a definujeme rozhraní USB,
3. nainstalujeme ovladače osciloskopu,
4. určíme rozhraní Interface Object (standard VISA),
5. přiřadíme ovladač osciloskopu k rozhraní,
6. zahájíme komunikaci se zařízením (s osciloskopem Tektronix).

Po navázání komunikace se zapsáním povelu `tektronix` v příkazovém řádku MMATLABu spustí okno GUI pro ovládání osciloskopu (viz obrázek č. 9 na straně 18). Zobrazení průběhu

[3] podrobný popis je uveden v kapitole 1. 3 této práce

měřené veličiny v okně GUI lze provést dvěma způsoby. Lze tak učinit stiskem tlačítka „Autoset“ (dojde k automatickému nastavení parametrů osciloskopu, k jednorázovému načtení dat a jejich vykreslení při optimalizaci velličiny pro zobrazení) nebo stiskem tlačítka „Run/Stop“. Po stlačení tlačítka „Run/Stop“ dojde k průběžnému plnění matice vstupních dat měřenými daty a jejich průběžnému vykreslováni na obrazovku. Po zaplnění všech řádků matice dat (počet řádků je definován pomocnou proměnnou `buffer` v příslušném editačním poli) dojde k průběžnému přepisování nejstarších dat daty novými. Ukončení vykreslování měřené veličiny provedeme opětovným stiskem tlačítka „Run/Stop“, přičemž v okně GUI zůstanou vykreslena poslední naměřená data.

V průběhu zobrazování měřené veličiny (a nejen při něm) je možno měnit základní nastavení osciloskopu – hodnotu časové základny, hodnotu rozlišení zobrazení (V/dílek), hodnotu násobku měřící sondy, výpočítat a zobrazit průběh FFT apod. Pokud tlačítko „Run/Stop“ není stisknuto, je po každé změně nastavení osciloskopu překreslen poslední zobrazený průběh na průběh s novými parametry.

Z načtených dat je možno vykreslit do samostatného okna stiskem tlačítka „FFT“ rychlou Fourierovu transformaci, respektive stiskem tlačítka „Spekt“ spektrogram signálu. Po opětovném stisku příslušného tlačítka dojde k vymazání zobrazeného průběhu. S ohledem na skutečnost, že vykreslovaný průběh FFT a spektrogram se zobrazuje v relativně malém okně, může uživatel po stisku tlačítka myši na příslušný průběh tento průběh otevřít v samostatném okně (figure FFT nebo figure Spektrogram – viz obrázek č. 10 a 11 na straně 20 a 21) a dále pak s tímto zobrazením pracovat.

Naměřená data je možno z grafického rozhraní exportovat dle požadavku uživatele. Data je možno uložit do výstupního souboru s požadovaným jménem nebo jejich průběh vytisknout. Výchozím formátem pro tisk průběhu měřené veličiny, zobrazené v okně GUI, je obrazový soubor ve formátu `jpeg`, který je uložený v pracovním adresáři programu MATLAB. Ukončení práce s grafickým uživatelským rozhraním je možno provést stiskem tlačítka „Close“, po němž dojde uzavření všech aktivních oken.

4. Závěr

Cílem této práce bylo vytvoření grafického uživatelského rozhraní v prostředí MATLAB pro navázání komunikace s osciloskopem Tektronix, získání a následné zpracování dat. Vytvořené grafické rozhraní dokáže věrně zobrazit průběhy ustálených nebo pomalu měnících se signálů, ovládat základní vlastnosti osciloskopu (nastavení časové základny, volba kanálu pro měření, nastavení násobku měřicí sondy, volba formátu zobrazení), vypočítat a zobrazit rychlou Fourierovu transformaci signálu, díky možnosti zobrazit si spektrogram průběhu si lze i vytvořit představu o změně veličiny v průběhu sledovaného období. Dále umí GUI zobrazit informaci o efektivní hodnotě měřené veličiny a taktéž zobrazit rozdíl mezi minimální a maximální hodnotou průběhu (rozdíl špička–špička).

Přínosem této práce (vytvořeného GUI) je dle mého názoru:

- umožnění snadného ovládání vlastností osciloskopu Tektronix TDS 2002B prostřednictvím počítače,

- možnost měření a zobrazení průběhu měřené veličiny prostřednictvím GUI i pro uživatele neznalého práce s osciloskopem (jednoduché spuštění aplikace, po stisku tlačítka „Autoset“ je v panelu GUI vykreslen průběh veličiny, osciloskop je nastaven pro optimalizované zobrazení měřeného průběhu. Dále jsou v infopanelu GUI zobrazeny údaje o rozdílu maxima a minima měřeného průběhu a také efektivní hodnota měřené veličiny),

- možnost měření (monitorování veličiny) prostřednictvím internetu. Osciloskop je možno ovládat například připojením k vzdálené ploše, které je k dispozici od verze operačního systému Windows XP nebo například pomocí VNC vieweru a VNC serveru,

- umožnění přenosu souboru s naměřenými daty, zobrazením naměřeného průběhu prostřednictvím internetu, aniž by se uživatel musel fyzicky připojit k osciloskopu.

Zmínil-li jsem v předchozím odstavci přínos této práce, musím v rámci objektivity upozornit na nedostatek, který se při používání GUI může projevit. Za nedostatek, limitující využitelnost GUI, považuji skutečnost, že měřit prostřednitvím grafického uživatelského rozhraní lze pouze ustálené nebo pomalu se měnící veličiny. U skokově se měnících veličin může dojít ke zkreslení mezi měřeným a zobrazeným průběhem. Případné zkreslení může být způsobeno například prodlevou při načítání dat (propojení počítače a osciloskopu USB kabelem) nebo prodlevou při komunikaci počítače s osciloskopem v důsledku zpracování naměřených dat (např. výpočet FFT, operace s maticemi dat při jejich zaplnění).

Použitá literatura

[1] *Www.tek2.com* [online]. 4. 10. 2006 [cit. 2010-04-13]. TDS200, TDS1000/TDS2000, TDS1000B/TDS2000B, and TPS2000 Programmer Digital Storage Oscilloscopes. Dostupné z WWW: <http://www2.tek.com/cmswpt/madetails.lotr?ct=MA&cs=mpm&ci=8784&lc=EN>.

[2] *Www.scopeshop.de* [online]. 9. 6. 2008 [cit. 2010-04-13]. Getting Started Guide Using MATLAB with Tektronix Oscilloscopes Over a USB Connection. Dostupné z WWW: <http://www.scopeshop.de/(S(n4zip3adbkwgdw3joapqvl45))/PRODUKTINFO/Oszilloskope/TDS2000B/File_2174/Matlab%20Instrument%20Driver.pdf>.

[3] ZAPLATÍLEK, K; DOŇAR, B. *Matlab – tvorba uživatelských aplikací*. 1.vyd. Praha : BEN, 2004. 215 s. ISBN 80-7300-133-0.

[4] *Www.mathworks.com* [online]. © 1994-2010 [cit. 2010-04-13]. Webové stránky firmy MathWorks. Dostupné z WWW: <http://www.mathworks.com/tektronix>.

[5] *Www.humusoft.cz* [online]. © HUMUSOFT 1991 - 2010 [cit. 2010-04-13]. Webové stránky obchodní společnosti HUMUSOFT s.r.o. . Dostupné z WWW: <http://www.humusoft.cz/produkty/matlab/matlab/>.

[6] *Www.conrad.com* [online]. © 2008 - 2009 [cit. 2010-04-13]. Download centrum firmy Conrad Electronic Česká republika, s.r.o. . Dostupné z WWW: <http://www2.produktinfo.conrad.com/datenblaetter/100000-124999/122854-an-01-cs-osciloskop.pdf>.

Příloha

Zdrojový kód programu je uložen v souboru `tektronix.m`, grafická podoba GUI je uložena v souboru `tektronix.fig`. Oba soubory spolu s ovladačem firmy MathWorks pro osciloskop Tektronix TDS 2002B v prostředí MATLAB tvoří přílohu tištěné verze této práce a jsou umístěny na samostatném nosiči CD ROM.